НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

«До захисту допущено»
Завідувач кафедри
_____ В.П. Тарасенко
(підпис)
"___"_____2019р.

Дипломний проект

освітньо-кваліфікаційного рівня "Бакалавр"

з напряму підготовки 6.050102 "Комп'ютерна інженерія"

на тему: «ПРОГРАМА АВТОМАТИЗАЦІЇ ГЕНЕРАЦІЇ МЕТАДАНИХ АУДІОФАЙЛІВ»

Виконав: студент 4 курсу, групи КВ-53
Мурдза Оксана Олегівна _____
(підпис)

Керівник асистент каф. СПіСКС Радченко К.О. _____
(підпис)

Консультант з нормоконтролю доц., к.т.н. Клятченко Я.М. _____
(підпис)

Рецензент доц., к.т.н. Марковський О.П. _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет прикладної математики

Кафедра системного програмування та спеціалізованих комп’ютерних систем

Освітньо-кваліфікаційний рівень “Бакалавр”

Напрям підготовки 6.050102 “Комп’ютерна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ В.П. Тарасенко
“___” _____ 2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Мурдзі Оксані Олегівні

1. Тема проекту «ПРОГРАМА АВТОМАТИЗАЦІЇ ГЕНЕРАЦІЇ МЕТАДАНИХ АУДІОФАЙЛІВ»,
керівник проекту Радченко Костянтин Олександрович, асистент каф. СПіСКС,
затверджені наказом по університету від “22” травня 2019 року № 1330-С

2. Строк подання студентом проекту: “___” _____ 2019 р.

3. Вихідні дані для дипломного проектування: див. Технічне завдання.

4. Перелік задач, які потрібно вирішити:
   - розробити програму обробки метаданих аудіофайлів;
   - забезпечити роботу з файловою системою користувача;
   - забезпечити автоматичну генерацію метаданих аудіофайлів;
   - підтримку багатомовного інтерфейсу користувача;
   - виконати тестування програми.

5. Перелік обов’язкового ілюстративного матеріалу:
   - структура обробки метаданих аудіофайла.
   - класи роботи з генерацією метаданих у аудіофайлах.
   - взаємозв'язок між класами системи (креслення).
   - алгоритм конвертації тегів в ім’я файлу.

6. Консультанти:

Питання	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М.		

7. Дата видачі завдання: “___” _____ 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	15.04.2019
2.	Розроблення та узгодження технічного завдання	30.04.2019
3.	Аналіз існуючих рішень	05.05.2019
4.	Підготовка матеріалів першого розділу дипломного проекту	10.05.2019
5.	Підготовка матеріалів другого розділу дипломного проекту	18.05.2019
6.	Підготовка графічної частини дипломного проекту	20.05.2019
7.	Оформлення документації дипломного проекту	25.05.2019
8.	Попередній огляд матеріалів диплому на кафедрі	30.05.2019

Студент _____ Мурдза О.О.

Керівник проекту _____ Радченко К.О.

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (___ с., ___ рис., ___ табл., ___ додатки).

Об’єкт розробки – розробка комп’ютерної програми автоматизації генерації метаданих у аудіо файлів.

Даний програмний комплекс створений для зручного та комфортного управління колекціями аудіофайлів на комп’ютері. До нього входять наступні елементи:

- програмний компонент, що відповідає за роботу з файловою системою користувача;
- програмний компонент, що забезпечує генерацію метаданих у аудіофайлах;
- програмний компонент, що відповідає за взаємозв'язок з між метаданими та відносним шляхом аудіофайлів у бібліотеці;
- простий веб-інтерфейс, що забезпечує можливість використовувати програму звичайному користувачу.

Програмний комплекс надає можливість легко керувати та обробляти аудіофайли на комп'ютері. Він дозволяє розбивати колекції аудіофайлів на логічні групи, генерує метадані, створює та обробляє групові метадані.

Ключові слова: метадані, MP3-файл, аудіофайли, управління колекціями, файлова система користувача.

ABSTRACT

Qualification work includes explanation note (___ p., ___ pic., ___ tab., ___ additions).

Object of development - the creation of a computer program to automate the generation of metadata in audio files.

This software package is designed for convenient and comfortable management of collections of audio files on a computer. It includes the following elements:

- the software component responsible for working with the user's file system;
- a software component that generates metadata in audio files;
- the software component responsible for the relationship between the metadata and relative audio files in the library;
- a simple web interface that provides the ability to use the program to the average user.

The software package provides an easy way to manage and handle audio files on your computer. It allows you to break the collection of audio files into logical groups, generate metadata, create and process group metadata.

Keywords: metadata, audio files, MP3-file, collections management, user file system.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045490.002 ТЗ	Програма авторизації	3		
			генерації метаданих			
			аудіофайлів			
			Технічне завдання			
	А4	ІАЛЦ.045490.003 ТП	Програма авторизації	2		
			генерації метаданих			
			аудіофайлів			
			Відомість проекту			
	А4	ІАЛЦ.045490.004 ПЗ	Програма авторизації	57		
			генерації метаданих			
			аудіофайлів			
			Пояснювальна записка			
	А4	ІАЛЦ.045490.005 Д1	Структура обробки	1		
			метаданих аудіофайла			
			Схема структурна			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.001 ОА		
Розроб.		Мурдза О.О.			Програма генерації метаданих аудіофайлів **Опис альбому**	Літ.	Аркуш 1 / Аркушів 2
Перевір.		Радченко К.О.					
Н. контр.		Клятченко Я.М.				КПІ імені Ігоря Сікорського, ФПМ КВ-53	
Затвер.		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.006 Д2	Алгоритм конвертації	1		
			тегів в ім’я файлу			
			Схема алгоритму			
	А4	ІАЛЦ.045490.007 Д3	Класи роботи з	1		
			генерацією метаданих			
			у аудіофайлах			
			Схема структурна			
	А4	ІАЛЦ.045490.008 Д4	Схема зв’язків класів	1		
			розробленої програми			
			Схема структурна			
		Диск CD-ROM	Матеріали дипломного			
			проекту.			
			Графічний матеріал			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.001 ОА	Арк. 2

ЗМІСТ

стор.

					ІАЛЦ.045490.002 ТЗ		
Зм.	Арк.	№ докум.	Підп.	Дата			
Розроб.		Мурдза О.О.			Програма автоматизації генерації метаданих аудіофайлів **Технічне завдання**	Літ.	Аркуш 1 / Аркушів 3
Перевір.		Радченко К.О.					
Н. контр.		Клятченко Я.М.				КПІ ім.Ігоря.Сікорського, ФПМ КВ-53	
Затв.		Тарасенко В.П.					

1. НАЙМЕНУВАННЯ І ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування роботи – «Програма автоматизації генерації метаданих аудіофайлів».

Область застосування: потокова обробка колекцій медіафайлів.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. ЦІЛЬ І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення програми, яка б забезпечила швидке створення, зміну, обробку логічної групи аудіофайлів.

4. ДЖЕРЕЛА РОБОТИ

Джерелами роботи є дослідні статті та книги на тему розробки графічного інтерфейсу, статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до програмного забезпечення:

- Операційна система Windows, Linux, OS.
- Java Development Kit.
- Пристрої виведення звуку(колонки, навушники).

5.2 Вимоги до апаратного забезпечення:

- Монітор.
- Комп’ютер
- Звукова карта.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	15.04.2019
2.	Розроблення та узгодження технічного завдання	30.04.2019
3.	Аналіз існуючих рішень	05.05.2019
4.	Підготовка матеріалів першого розділу дипломного проекту	10.05.2019
5.	Підготовка матеріалів другого розділу дипломного проекту	18.05.2019
6.	Підготовка графічної частини дипломного проекту	20.05.2019
7.	Оформлення документації дипломного проекту	25.05.2019
8.	Попередній огляд матеріалів диплому на кафедрі	30.05.2019

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045490.004 ПЗ	Програма авторизації	57		
			генерації метаданих			
			аудіофайлів			
			Пояснювальна записка			
	А4	ІАЛЦ.045490.005 Д1	Структура обробки	1		
			метаданих аудіофайла			
			Схема структурна			
	А4	ІАЛЦ.045490.006 Д2	Алгоритм конвертації	1		
			тегів в ім’я файлу			
			Схема алгоритму			
	А4	ІАЛЦ.045490.007 Д3	Класи роботи з	1		
			генерацією метаданих			
			у аудіофайлах			
			Схема структурна			
	А4	ІАЛЦ.045490.008 Д4	Схема зв’язків класів	1		
			розробленої програми			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.003 ТП		
Розроб.		Мурдза О.О.			Програма автоматизації генерації метаданих аудіофайлів **Відомість дипломного проекту**	Літ.	Аркуш 1 / Аркушів 2
Перевір.		Радченко К.О.					
Н. контр.		Клятченко Я.М.				КПІ імені Ігоря Сікорського, ФПМ КВ-53	
Затвер.		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
		Диск CD-ROM	Матеріали дипломного			
			проекту.			
			Графічний матеріал			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.003 ТП	Арк. 2

ЗМІСТ

стор.

ДОДАТКИ

Додаток 1. Копії графічного матеріалу

ІАЛЦ.467200.005 Д1. Структура обробки метеданих аудіофайла.

ІАЛЦ.467200.006 Д2. Алгоритм конвертації тегів в ім’я файлу.

ІАЛЦ.467200.007 Д3. Класи роботи з генерацією метаданих у аудіофайлах.

ІАЛЦ.467200.008 Д4. Схема зв’язків класів розробленої програми

Додаток 2. Лістинг програми

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ОС – операційна система.

Kotlin, Java – це мови програмування сімейства JVM.

Інтернет – це велика всесвітня розгалужена мережа, яка об’єднує безліч людей логічно пов’язаними комп’ютерними вузлами.

Гаджет – це електронний пристрій, який пов'язаний з інтернетом.

Аудіофайл – це це звуковий файл, який містить відцифровані звукові хвилі. Далі детальніше розглянемо особливості аудіофайли формату MP3.

Фреймворк – це структура у вигляді певного шаблону з реалізацією певного базового коду, зазвичай прийнято називати бібліотеками.

MP3 – це формат аудіофайлів для зберігання та передавання інформації.

WAV (waveform audio format) – формат для аудіофайлів, який створим компанія Microsoft.

AIFF (Audio Interchange File Format) – це формат аудіофайлів для електронних пристроїв.

AU – це простий формат для аудіофайлів представлений компанією Sun Microsystems.

PCM (Pulse Code Modulation) – імпульсно-кодова модуляція, процес перетворення з аналогового в цифровий сигнал.

Кодек – це пристрій або програма шифровки/дешифровки.

FLAF (Free Lossless Audio Codec) — вільний аудіокодек без втрат.

.m4a – одне з розширень аудіофайлів на пристроях компанії Apple, може бути перейменованим в .mp4.

DCMI – один з найпоширеніших форматів опису метаданих для файлів будь-якого типу.

vCard (Virtual Contact File) – стандартний формат файлів.

WMA - lossy Windows Media Audio.

MARC (Machine-Readable Cataloging) – формат машиночитного каталогізаційного запису.

ONIX (ONline Information eXchange) – це сімейство форматів XML.

Ogg Vorbis (ogg) – вільний формат стиснення звуку для аудіофалів.

Musepack – неліцензований формат файлу для зберігання аудіо, який є одним з найпопулярніших форматів.

AAC (Advanced Audio Coding) – стандартна схема стиснення із втратами для аудіоданих.

ATRAC (Adaptive Transform Acoustic Coding) - сімейство аудіо алгоритмів стиснення, розроблених компанією Sony.

FoaF - проект зі створення моделей машинно-читаних домашніх сторінок і соціальних мереж.

ID3 (Identify a MP3) – специфікація для MP3-файлів

Junit – це фреймворк для створення тестів програмного забезпечення, він використовується для мови програмування Java.

ВСТУП

На сьогоднішній день активно розвивається комунікація через Інтернет. Більшість людей багато часу проводять з гаджетами. Вони переглядають актуальні новини, розважальні відео, погоду на день, повідомлення та листи, які приходять через соціальні мережі чи електронну. Під час ходьби на роботу, пробіжки зранку та інших ситуацій, люди слухають бадьору музику. Складно уявити на сьогоднішній день життя без обміну даними через Інтернет. Щоденно люди взнають безліч інформації через нього, тому, відповідно, збільшується попит на передачу даних, з мінімальними затратами мобільного трафіку, швидкою та надійною передачею інформації через Інтернет.

Можна стверджувати, що кожен , хто любить слухати музику, стикався з тим, що при відтворенні її через музичний плеєр відображаються не зрозумілі символи, замість назви, виконавця, чи альбому або взагалі не вказана інформація про композиції.

Кожен має колекцію улюблених музичних композицій, але, частіше всього, вони мають різні формати, не завжди ім’я виконавця є правильно написаним або обкладинка альбому однаково вказана для всіх композицій. У більшості пісень немає вказаного тексту композиції, а деколи так хотілось би їх прочитати. Цю проблему виправити не так просто, як перейменувати файл.

Більшість аудіофайлів мають метадані, саме вони містять у собі теги з назвою твору, альбому, автора. Аудіофайли та метадані бувають різних форматів, саму тому ці дані не можливо виправити вручну без додаткових програм.

Ця проблема є актуальною, так як програм, які можуть допомогти з вирішенням цієї проблеми, є не так багато. У всіх програм, що працюють з обробкою аудіофайлів є свої недоліки та переваги. Вони створюються для різних типів даних, задач, різних операційних систем та відповідно до інші вимог. Переважно вони реалізовані для професійної обробки аудіофайлів та

являються платними. Також недоліків багатьох є те що вони виконують не всі потрібні функції, або не зручні у використанні, тому потреба у якісному продукті лишається.

Оскільки, на даний момент майже кожен має колекції аудіофайлів та й загалом аудіофайли досить часто використовують люди, тому виникає необхідність у програмах, які б допомагали легко та швидко управляти аудіофалами. Ця галузь лишається актуальною на даний момент, тому появляються нові програми та оновлені версії.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ, ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1 Теоретичні відомості

На сьогодні існують різні види аудіофайлів та методи обробки їх метаданих. Розглянемо аудіофайли, їх види структуру та обробку.

Існує багата різних цифрових звукових форматів, які використовуються на даний момент. Можна виділити 3 основні групи цифрових звукових форматів:

- нестиснені формати — такі як WAV, AIFF, AU або PCM;
- формати із стисненням без втрат — FLAC, WMA Lossless, Monkeys` audio, Apple Lossless [1].
- формати із стисненням з втратами – такі як наприклад MP3, Ogg Vorbis, Musepack, AAC, ATRAC чи WMA.

Нестиснені формати – це різні формати аудіофайлів, до яких не застосовується стиснення розміру файлу. Переважно в таких форматак файлів виділяється мала кількість пам’яті на метадані. Такі формати є широко розповсюдженими та підтримуються на даний момент [1].

Формати із стисненням з втратами – надає можливість набагато більшого стиснення файлу завдяки видаленню частини аудіоінформації та спростивши дані. Це, звичайно, призводить до зниження якості звуку, але використовуються різноманітні методи відновлення.

Формати із стисненням без втрат зберігають дані на меншому просторі без втрати інформації. Також стиснені файли можна відновити до оригінального розміру за допомогою додаткових програм.

На даний момент існує безліч форматів аудіофайлів на ринку, вони відрізняються своїми характеристиками. Проаналізуємо основні формати аудіофайлів.

Назва формату	Квантування, біт	Число каналів	Величина потоку даних з диску, кбіт/с	Ступінь стиснення
MP3	16	2	128 (12-320)	~11:1 з втратами
Ogg Vorbis	до 32	до 255	не обмежений	~22:1 (при 64 kbps) з втратами
CD	16	2	1411,2	1:1 без втрат
Dolby Digital 5.1	16-24	6	448	~12:1 з втратами
DTS	20-24	6	більше 768	~7:1 з втратами
DVD-Audio	24	6	6912	2:1 без втрат
MPEG Audio Layer III	плаваючий	2	до 320	~11:1 з втратами
MPEG AAC		до 48	до 529 (стерео)	з втратами
WMA	до 24	до 8	до 768	2:1, є версія без стиснення

Табл. 1. Формати аудіофайли та їх основні характеристики

В таблиці 1 представлені найпопулярніші формати аудіофайлів. В залежності від потреб використовуються різні формати зберігання файлів [2].

MP3 формат представлений для зберігання аудіофайлів та є найбільш розповсюдженим. На відміну від популярного аналога CD формату, MP3 має малий розмір, тому зручний для пересилання . Якщо порівнювати їх ступінь стиснення, то CD формат не має стиснення. MP3 має ступінь стиснення 11:1 і за рахунок цього погіршується якість звучання. На даний момент для пересилання файлів та звичайного користування лишається популярним MP3 формат. CD формат дозволяє записати файл більшого обсягу, що робить якість музики ще краще. DVD-Audio також є популярним, він рідше використовується ніж CD та MP3 [4]. Йому надають перевагу любителі якісної музики, які мають спеціальне обладнання для прослуховування, але

зазвичай пристрої відтворення аудіофайлів не мають потрібних характеристик для відтворення з відчутним ефектом.

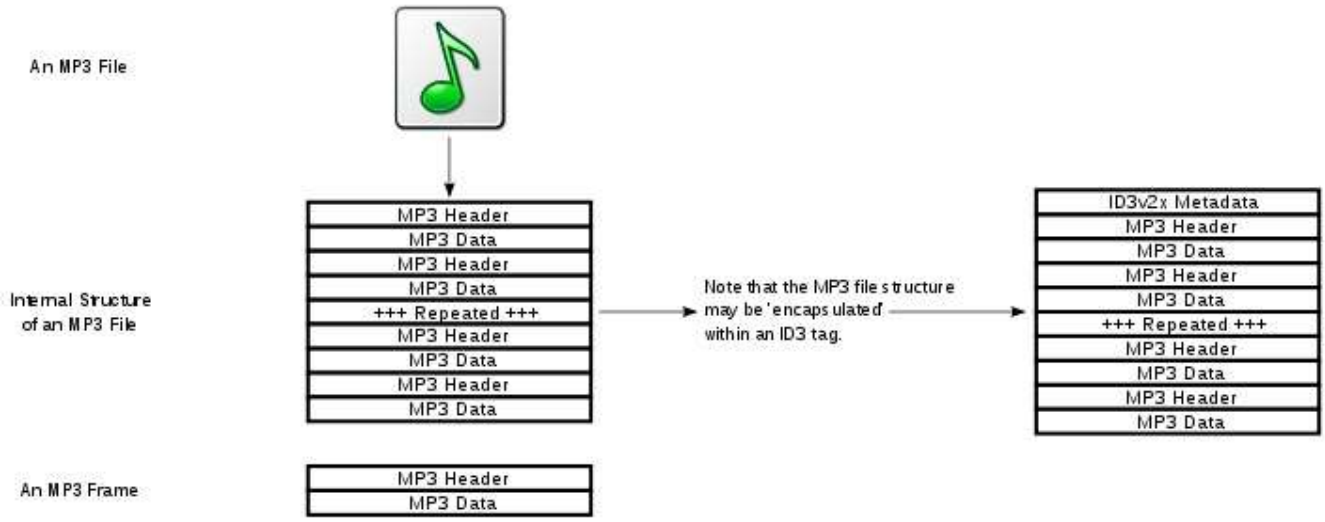

Рис. 1. Структура MP3-фалу

Розглянемо структуру MP3-фалу. З рисунку 1 наглядно видно, що MP3-файл складається із впорядкованих кадрів, розподілених на дві частини: заголовок та блок даних. Кадри формують елементарний потік і є залежними частинами, тому не можна вилучити окремий кадр.

У більшості аудіофайлах, на даний момент часу, є метадані. Метадані– це впорядковані дані, які характеризують задану систему даних. Метадані знаходяться в окремому розділі та містить інформацію про заголовок, автора, виконавця, рік та назву альбому, номер доріжки та іншу інформацію про вміст звукового файлу. На схемі з додатку 1 зображений блок метаданих для MP3-файлу [3]. Цей блок може містити тільки частину з представлених даних або взагалі не мати метаданих.

Метадані класифікують:

- за змістом;
- за приналежністю до ресурсу:
  - до всього ресурсу;
  - до його частини.

- за логічним виводом:
    - нижнього шару;
    - середнього шару;
    - верхнього шару.

Існують різні формати метаданих, такі як: DCMI, vCard, FOAF, MARC, ONIX, CIF. Розглянемо детальніше ID3, він найчастіше використовується для популярних форматів, таких як MP3 [5].

Поле	Довжина	Опис
заголовка	3	"TAG"
назва	30	30 символів заголовка
художник	30	30 символів імені виконавця
альбом	30	30 символів назви альбому
рік	4	Чотиризначний рік
коментар	28 або 30	Коментар.
нульовий байт	1	Якщо номер доріжки зберігається, цей байт містить двійковий номер 0.
доріжка	1	Номер треку в альбомі або 0. Недійсний, якщо попередній байт не є двійковим 0.
жанру	1	Індекс у списку жанрів або 255

Табл. 2 Формат ID3v1.1

З представлених даних на таблиці 2 можем побачити, що метадані ID3v1 займають 128 байт. Вони розміщуються після даних, у кінці файлу, щоб підтримувати сумісність зі старими медіаплеєрами. Аналізуючи стовпчик з назвою «довжина» ми бачимо, що виділено по 30 байт для назви аудіофайлу, виконавця, альбому та додаткової інформації, чотири байти на рік і байт для ідентифікації жанру пісні з наперед визначеного списку 80-ти значень [6].

Поле	Довжина	Опис
заголовка	4	"TAG +"
назва	60	60 символів заголовка
художник	60	60 символів імені виконавця
альбом	60	60 символів назви альбому
швидкість	1	0 = не встановлено, 1 = повільно, 2 = середнє, 3 = швидко, 4 = хардкор
жанру	30	Вільне текстове поле для жанру
Час початку	6	початок музики формату: mmm: ss
кінцевий час	6	кінець музики формату: mmm: ss

Табл. 3 Формат ID3v1.2

Через деякий час, тег ID3v1 видозмінили, ключовим моментом стало його розширення, а саме збільшення виділяємої пам’яті на кожне поле даних та зміна деяких полів. Порівнюючи таблицю 2 та таблицю 3, бачимо, що зросла довжина заголовка на 1 байт, назви, виконавця, альбому – на 30

символів, поле жанру замінили на текстові дані та добавили і забрали деякі поля даних.

Тег ID3v2 хоч містить ID3, але її структура сильно відрізняється від попередньої. Вони мають змінний розмір та переважно використовуються на початку аудіофайлу. Це досить зручно, як тільки почнеться потокова загрузка файлу, метадані стануть доступні. Ця версія не вимагає повної загрузки файлу перед обробкою метаданих [4].

Рис. 2. Заголовок ID3v2

На даний момент є 3 версії: ID3v2.2, ID3v2.3 і ID3v2.4. Також у версіях ID3v2 при вказуванні довжини даних кожен сьомий біт встановлюється в 0 і не використовується, це продемонстровано на рис.3. У ній також добавили можливість закріплення обкладинка альбому та можливість вказати формату зображення [9].

Загалом, теги ID3 були розроблені з урахуванням MP3, тому вони добре взаємодіють з файлами MP3 і MP3Pro . Однак, теги є незалежною частиною, а не частиною файлу MP3 і повинні бути використані в інших місцях. На практиці, єдиними іншими форматами, які широко використовують теги ID3v2, є AIFF і WAV .

Специфікація кадру ID3v2 в десятки раз більша. Якщо у кадрі ID3v1 представлено 7-8 полів з метаданими (в залежності від версії), то у ID3v2 приблизно 100 полів [7].

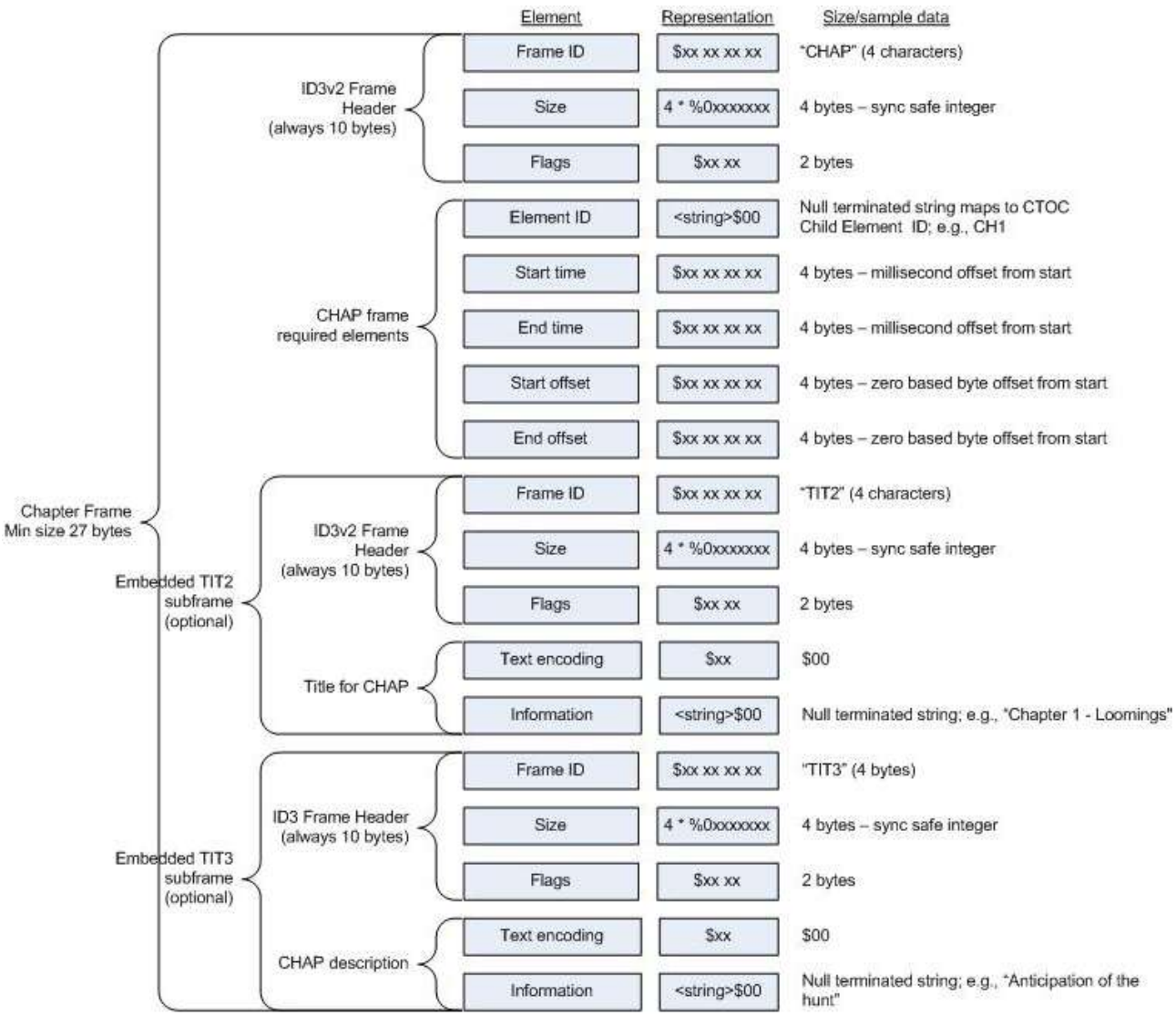

Рис. 3. Структура ID3v2

Таблиця з усіма даними формату ID3v2 дуже велика, тому представлений узагальнений розподіл полів на рисунку 4. Порівнюючи таблиці 2 і 3 з рисунком 4, можна побачити, що обсяг даних зріс та сильно змінився [10]. Якщо розгляну співвідношення довжин цих двох специфікацій, то маємо наступні дані:

ID3v1:

- поля мають різну довжину зображені на рисунках 2 та 3;
- загальна довжина – 128 байт;
- виділена пам’ять під кожне поле не змінна;

ID3v2:

- кожне поле може містити до 16 Мб,
- загальна довжина під усю кількість тегів (полів) до 256 Мб;
- в залежності від кількості використання полів виділена пам'ять може змінюватись.

Існує багато варіантів редагування тегів ID3. На спеціальних платформах можна редагувати метадані аудіофайлу, при перегляді через розширену версію у файловому менеджері або через спеціальні програми. Також деякі аудіопрогравачі дозволяють редагувати аудіофайл з мінімальним набором інструментів, але зазвичай його недостатньо.

1.2. Існуючі аналоги

На даний момент існують аналоги обробки метаданих аудіофайлів. Більшість редакторів орієнтовані на різну цільову аудиторію. Проаналізувавши багато програм я виділила дві основні категорії.

Перша категорія програм орієнтується на більш професійного користувача. Такі програми мають досить складний інтерфейс, але дають можливість складної і глибокої обробки, за рахунок великої кількості інструментів. Такі програми дозволяють обробку великої кількість форматів, зручну навігацію по файловій системі користувача. Якщо програма має досить велику кількість інструментів, то ускладнюється реалізація та розміщення їх в інтерфейсі. Також вони переважно мають свої особливі функції. Переважно вони мають пошук в інтернеті, конвертування таблиць спеціальних та інше.

Друга категорія програм базується на одноразовому використанні програми. мінімальному наборі інструментів та дозволяють по одиночне редагування аудіофайлів підтримка одного чи декількох форматів. Також такі програми переважно написані під одну ОС.

Переглянемо 5 найпопулярніших програм для обробки аудіофайлів та розберемось у їх перевагах та недоліках, цілі та доречність використання кожної з програм. Аналіз програм базується на власному досвіді та зборі інформації від користувачів даних програм та статей про них.

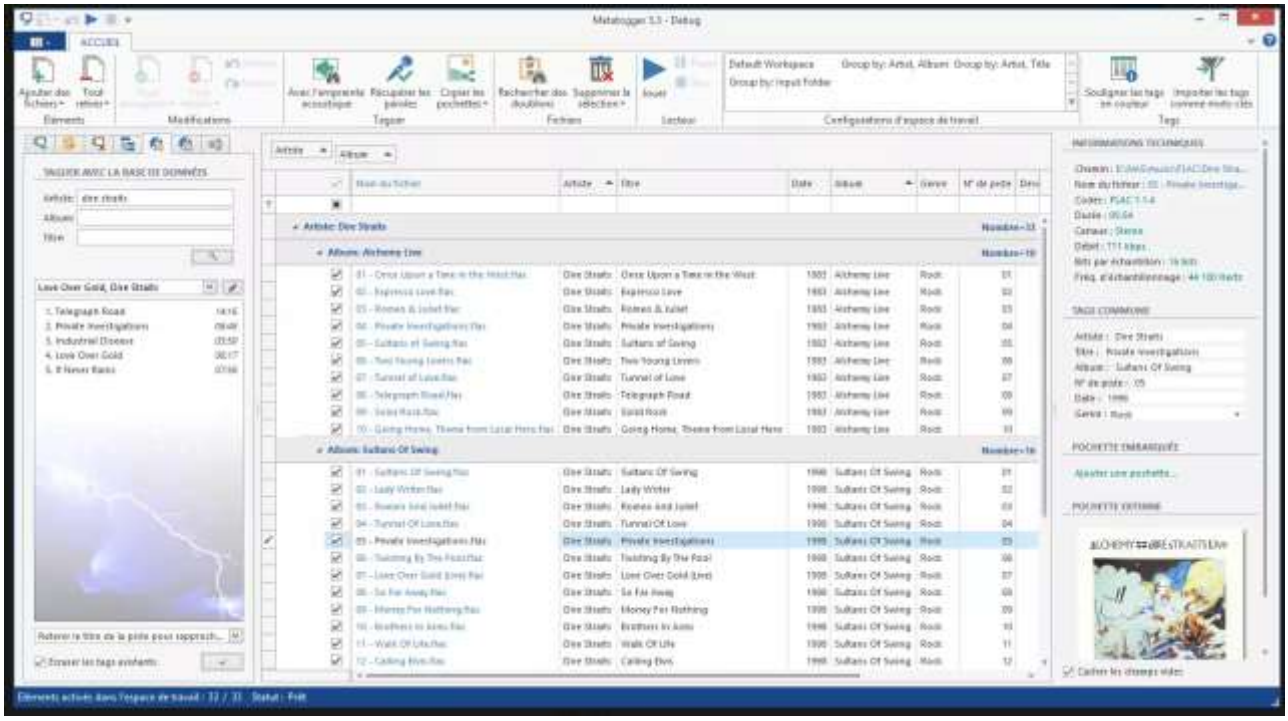

Рис. 4. Інтерфейс програми MetaTOGGer.

MetaTOGGer – використовується для обробки MP3-файлів. За допомогою цієї програми можна обробляти музичні файли вручну або за допомогою баз даних. Ця програма дозволяє пошук та скачування обкладинок альбомів зі спеціальних ресурсів. Програми використовує Microsoft.Net 3.5 фреймворк тому, щоб почати роботу з цією програмою доведеться спочатку скачати цю бібліотеку, якщо вона попередньо не була установлена на робочій системі. Також досить складно простому користувачу розібратись з інтерфейсом програми. Він містить велику кількість інструментів часто незручно розташованих, побачити інтерфейс програми можна побачити на рисунку 4. Ця програма працює з багатьма форматами файлів та може завантажувати потрібні дані з інтернету.

Переваги:

- робота з багатьма форматами файлів
- завантаження даних з Інтернету

Недоліки:

- без додаткових бібліотек не працює
- не зручний інтерфейс
- дозволяє тільки по одиночне редагування
- без підтримки багатомовності

TigoTago – це редактор тегів. Програма підтримує не тільки різні формати аудіофайлів, але й обробляє відеоформати. Вона може редагувати групи файлів та має необхідну базу користувацьких інструментів.

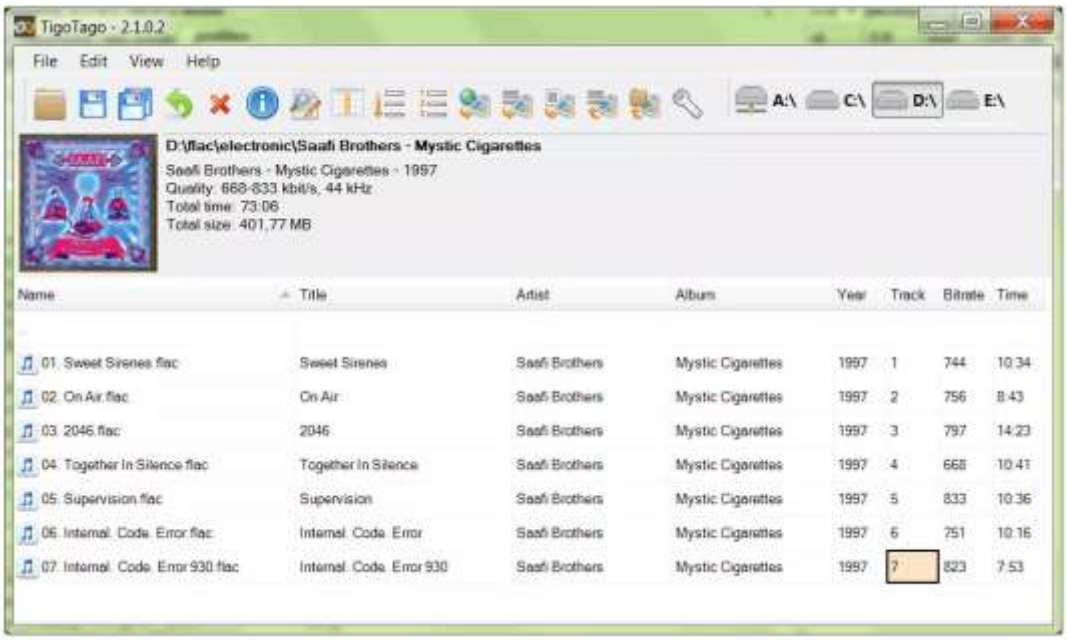

Рис. 5. Інтерфейс програми TigoTago.

У програми складно реалізований доступ до аудіофайлів та немає можливості роботи з файлової системою, немає багатомовної підтримки, але сам інтерфейс програми виконаний у мінімалістичному стилі, що візуально робить приємною програму. Програмний інтерфейс даної програми зображений на рисунку 5.

Переваги:

- підтримує різні формати
- дозволяє редагування відеофайлів
- редагування групи аудіофайлів

Недоліки:

- не можливість взаємодії з файловою системою
- немає підтримки багатомовності
- обробляє тільки теги аудіофайлів

MusicBrainz Picard – це музичний редактор, для редагування тегів, який працює на різних операційних системах. Він є безкоштовним та базується на обробці альбомів.

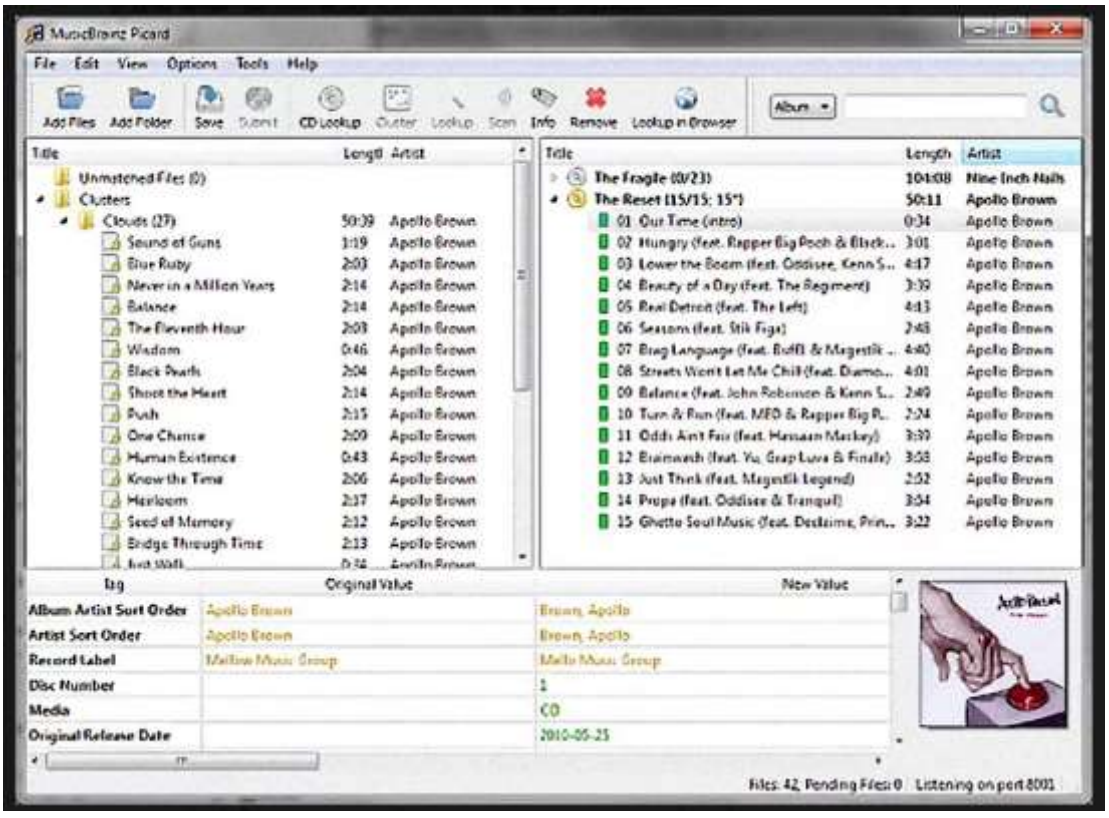

Рис. 6. Інтерфейс програми MusicBrainz Picard.

Інтерфейс програми на різних операційних системах доволі сильно відрізняється. На рисунку 6 зображений інтерфейс для ОС Windows. А також

програма включає в себе графік з кривою, яка відображає зміну швидкості навчання функцій запропонованих програмою. Графік не зручно розташований, часто збивається і працює некоректно.

Переваги:

- безкоштовна програма
- базується на обробці альбомів

Недоліки:

- різний інтерфейс на різних операційних системах
- графік кривизни навчання
- не зручний інтерфейс

TagScanner - це програма для операційної системи Windows, яка підтримує більшість музичних форматів. Працює з великою кількістю форматів, а також у ній є вбудований програвач.

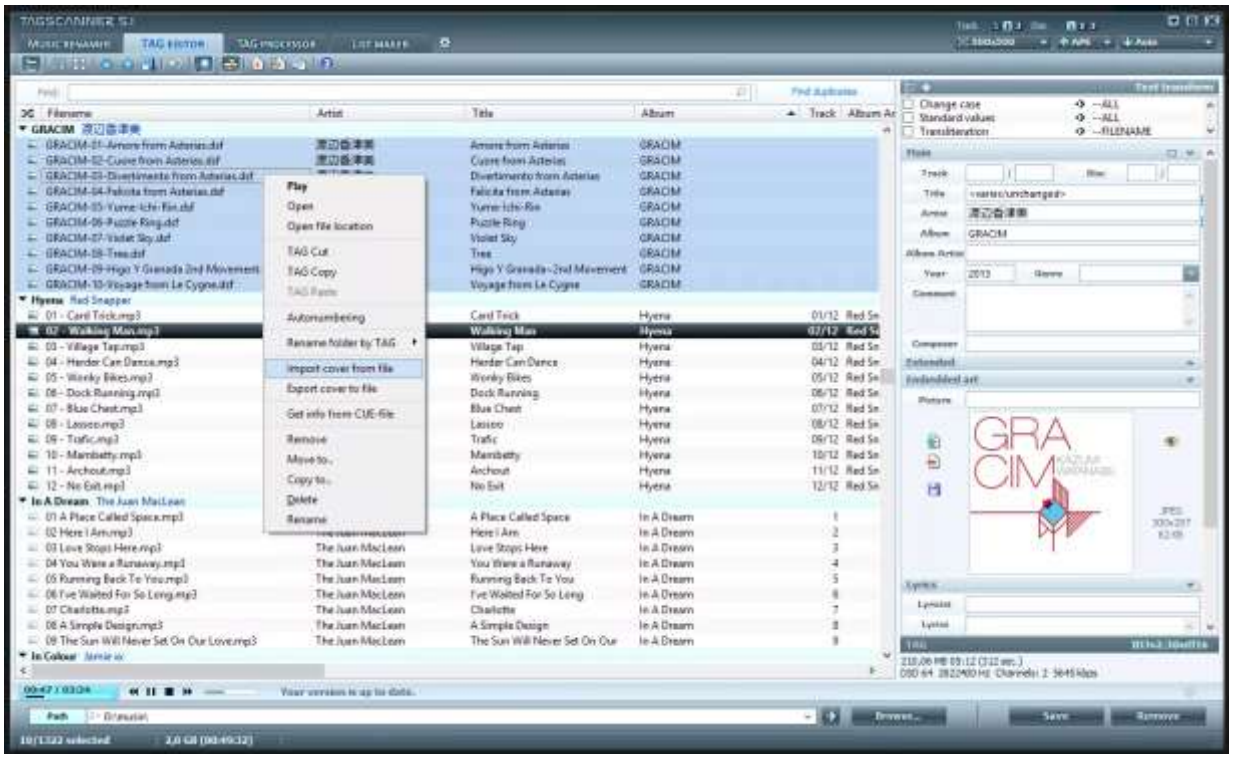

Рис. 7. Інтерфейс програми TagScanner.

Ця програма розроблена для більш професійного використання. Ця програма може автоматично заповнювати пусті метадані за допомогою

спеціальних онлайн баз даних. Також редактор може експортувати списки відтворення у вигляді таблиці HTML або Excel. Так як програма розроблена для професійної діяльності вона складна у використанні та має досить складний інтерфейс це можна побачити на рисунку 7.

Переваги:

- автоматично заповнює пусті метадані
- дозволяє експорт в таблиці HTML або Excel

Недоліки:

- складна програма
- без підтримки багатомовності
- для професійного користування
- заплутаний інтерфейс

MP3tag - це редактор метаданих, який підтримує тільки декілька форматів.

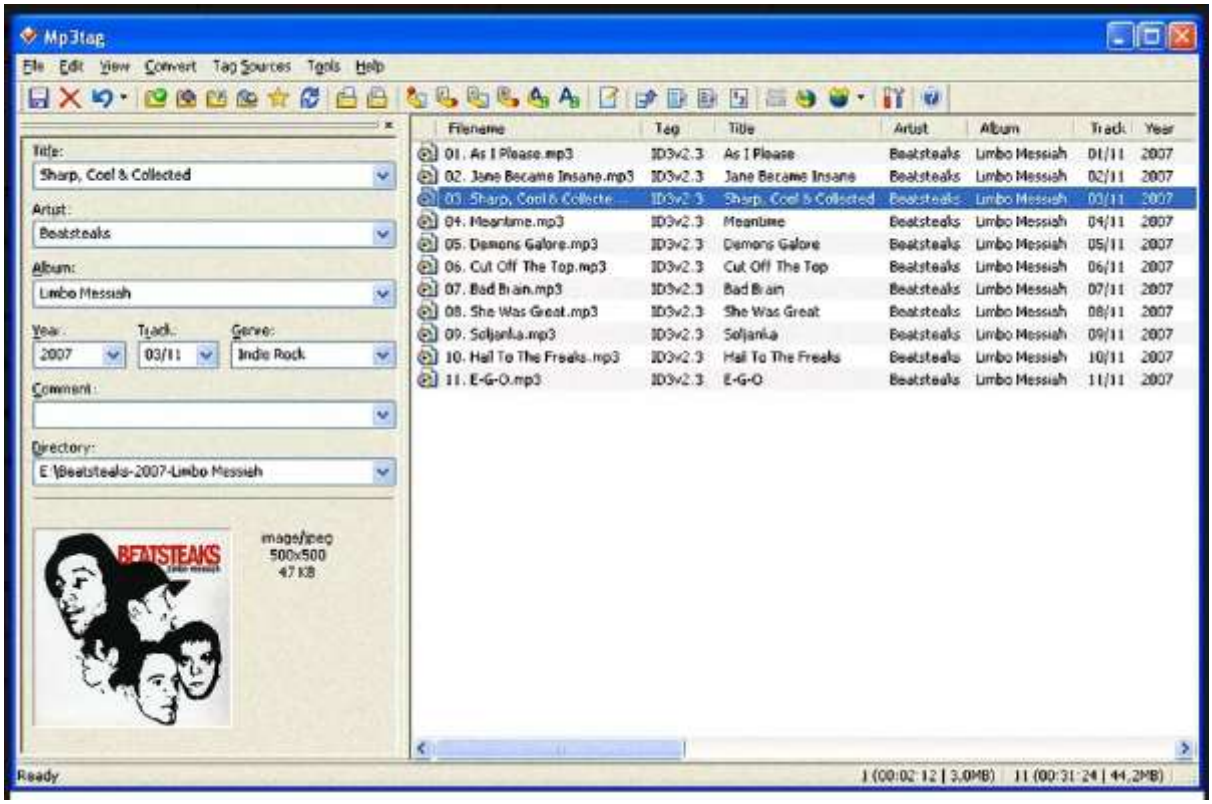

Рис. 8. Інтерфейс програми MP3tag

Досить зручним є те, що програма має онлайн пошук метаданих. Ця програма підтримує найпопулярніші формати, які підтримують стандарт формату метаданих ID3v2.

Досить зручно, що програма підтримує онлайн пошук метаданих, простий та зручний інтерфейс, що представлений на рисунку 8. Важливим недоліком цієї програми є те, що програма не зберігає зміни при закритті, потрібно не забувати натискати відповідну кнопку для зберігання змін.

Переваги:

- підтримує основні формати
- онлайн пошук метаданих
- простий та зручний інтерфейс

Недоліки:

- неможливість обробки групи аудіофайлів
- немає доступу до обробки файлової системи
- без підтримки багатомовності
- зміни не зберігаються при закритті

1.3. Обґрунтування теми дипломного проекту

Зазвичай з проблемою обробки аудіофайлів стикаються багато людей. Проаналізувавши популярні формати аудіофайлів, можна стверджувати, що найчастіше люди використовують файли MP3, m4a та flac, для обробки їх метаданих використовується формат ID3v2. Програм які обробляють метадані тільки конкретної групи аудіофайлів існує не так багато, а програми які оробляють багато форматів переважно складні і використовуються для професійного спрямування.

Аналізуючи усе перечислене вище, ми розумієм, що ця галузь лишається актуальною. Виходячи з цього темою дипломної роботи було

обрано програма автоматизації генерації метаданих у аудіофайлах. Також було виділено декілька основних пунктів для створення програми.

Інтерфейс повинен бути простим та зручним для користувача, мати багатомовну підтримку та змінні налаштування інтерфейсу для полегшення роботи.

Програма повинна містити не тільки редагування метаданих аудіофайлі, але й підтримку файлової системи користувача. Різні функції для відсортовування різних файлів, створення та мінімальний набір інструментів для роботи з папками. Також можливість пошуку та фільтраціїї за вибраними користувачем критеріями.

Основним пунктом є обробка аудіофайлів. Програма повинна містити обробку логічної групи метаданих аудіофайлів, обробка одного аудіофайлу, управління колекціями аудіофайлів.

Також програма повинна бути оптимізованою та затрачати мінімум системних ресурсів. Має бути досягнута підтримка на різних ОС та працювати коректно без додаткових бібліотек та фреймворків. Робота програми не повинна залежити від наявності Інтернету чи інших факторів.

2. РОЗРОБКА КОМПОНЕНТІВ СИСТЕМИ

2.1 Загальна структура системи

Структуру даної програми можна розбити на 4 основних частини:

- filechanger
- gui
- audiotag
- setting

Модуль filechanger – це реалізація зовнішнього інтерфейсу, який надає зручний та швидкий доступ для комунікації з користувачем. Він містить в собі методи для коректної роботи з відображенням файлової структури комп'ютерної системи користувача. У тому числі і для відображення роботи з метаданими аудіофайлів.

Модуль gui – це частина графічного інтерфейсу, яка служить основою для модуля filechanger. У ньому знаходиться реалізація кожної з частин зображення інтерфейсу як: вигляд панелі налаштувань, вікно з поточними файлами обробки та багато інших.

Модуль audiotag – це частина основним завданням є пряма робота з метаданими аудіофайлів. Він реалізує генерування імен, витягування тегів з аудіофайлів, генерує і досягає метадані для подальших обробок.

Модуль setting – відповідає за налаштування даної програми. У нього входять такі аспекти як: підтримка багатомовності, вибір та подання інформації, налаштування певних функції програми для зручного користування.

Модуль filechanger складається з:

- FileChanger.kt – це клас, що надає загальний інтерфейс для роботи з файловою системою користувача. Він дозволяє зручне управління

файлової системи та колекціями аудіофайлів. Детальніше функції управління будуть описані у відповіднних класах.

- FileChangerConteiner.kt – це клас, який представляє собою високорівневу обгортку над класом FileChanger.kt. Він реалізує більш зручний інтерфейс для обробки файлів та надає можливість потокової обробки файлів.

Модуль gui складається з:

- підмодуля toolbar. Цей підмодуль відповідає за реалізацію графічного інтерфейсу панелі інструментів та містить такі класи:
  - ConvertTab.kt- це клас, що відповідає за графічну реалізацію генерацію метаданих у аудіофайлах, на основі вибраного імені в файловій системі та навпаки – імені файлу з його метаданих.
  - FilesTab.kt – це клас, який являється компонентом панелі інструментів. Він відповідає за зображення роботи файлів в файловій системі такої як: навігація, створення, видалення папок та вибір одного чи декількох файлів для наступної обробки.
  - FilterTab.kt – це клас, що відповідає за візуалізацію пошуку по усім наявним файлам за заданими шаблонами або фільтрами
  - ReplaceTab.kt – це клас, який реалізує весь процес зв’язаний з перейменування, видаленням та перезавантаженням файлів
  - SettingsTab.kt – це клас, що відповідає за зображення вікна налаштувань проекту, його розмір, мову інтерфейсу, шрифт та інших додаткових налаштувань
  - TagsTab.kt – це інтерфейс, що дозволяє обробку метаданих аудіофайлів
  - ToolBar.kt – це клас, що реалізує зображення вікна зі списком поточних файлів та панелі налаштувань

- FileModel.kt – це клас, що реалізує модель поведінки файлової системи, для представлення структури даних у виді дерева
- FileTable.kt – це клас, що зображує таблицю з виділенням вибраних файлів в поточній папці, для подальшої обробки
- FileTree.kt – це клас, який реалізує логічне та зручне зображення даних для користувача файлової системи з дерева даних, яке обробляє програма у простий список
- MainVeiw.kt – це клас, що відповідає за головне вікно програми. У цьому вікні знаходяться усі компоненти графічного інтерфейсу.
- TableColumnManager.java – це клас, який реалізує компоненти для таблиці файлів. У ньому написані функції які дозволяють управляти колонками, рядками, повзунками, та інші, що роблять візуалізацію більш красивою та зручною.

Модуль audiotag складається з:

- TagCreator.kt – це клас, що відповідає за генерацію метаданих у аудіофайлах на основі його імені в файловій системі та навпаки – імені файлу з його метаданих. Генерація відбувається за допомогою стандартного алгоритму або шаблону, який задає користувач.
- TagReader.kt – це клас, який реалізує витягування тегів з MP3 файлу

Модуль settings складається з:

- Force.kt – клас, який відповідає за роботу з параметрами, що являються логічними змінними
- Language.kt – клас, що забезпечує підтримку багатомовного інтерфейсу користувача
- SaveLoad.kt – клас, що відповідає за спеціальні можливості користувача такі як: зберігання обробленої інформації, зберігання налаштувань при

перезапуску системи або повторному використання через деякий проміжок часу

- Settings.kt – це загальний клас, що реалізований для роботи налаштувань програми
- SizeResolver.kt – клас, що відповідає за роботу з параметрами, що являються числовими змінними

Тепер, після опису усіх модулів можна розглянути основні класи та їх реалізацію.

2.2. Модуль filechanger

Цей модуль складається з двох класів, розглянемо основні їх методи та продемонстуємо частинки коду.

2.2.1. Опис класу FileChanger

Цей клас надає загальний інтерфейс для роботи з файловою системою користувача. Він дозволяє зручне управління файлової системи та колекціями аудіофайлів. У нього такі основні методи: moveByName, move, capitalize, getReplacement, load. Розглянемо кожен з перечисленних методів.

Метод moveByName відповідає за перенесення файлу з однієї директорії в іншу за заданим ім’ям файлу. Нижче наведений код метода.

```
fun moveByName(file: File, path: String = "", from: String = "", to: String = " - ", force:
Boolean = true): File {
    if (from == "" && to == "")
        return file
    val name = file.nameWithoutExtension
    var start = name.indexOf(from)
    if (start != -1)
        start += from.length
    val end = name.indexOf(to)
    if (start == -1 && end == -1)
        return file
```

```
        val dir = when {
                to == "" -> name.subSequence(start, name.length)
                from == "" -> name.subSequence(0, end)
                else -> name.subSequence(start, end)
            } as String
        val destination = if (path.isEmpty()) "${file.absoluteFile.parentFile.absolutePath}/$dir"
else "$path/$dir"
        if (!File(destination).exists() && force)
            File(destination).mkdirs()
        val new = File("$destination$sep${file.name}")
        if (force)
            file.renameTo(new)
        return new
    }
```

Також у програмі реалізовані ще два аналогічних метода. Відмінність їх від попереднього методу у тому, що задається не ім'я, а у першому передається шлях до файлу. У другому передається сам файл. Так як Kotlin дозволяє реалізацію методів з однаковими іменами, надані сігнатури методів, код методів є досить схожим.

- move(file: File, path: String, force: Boolean = true): File
- move(file: File, dir: File, force: Boolean = true): File

Метод capitalize – змінює букви в імені файла з великих букв на малі, а після цього першу літеру робить великою. Приклад: «НАЗВА» - «Назва». Нижче представлений код програми.

```
    fun capitalize(file: File, delimiter: String, allBefore: Boolean, beforeDelim: String = " ",
force: Boolean): File {
        val dir = file.absoluteFile.parentFile.absolutePath
        val name = file.nameWithoutExtension
        val parts = name.split(delimiter)
        val before = parts[0]
        val after = parts.subList(1, parts.size).joinToString(separator = delimiter).capitalize()
        val newBefore =
            if (allBefore)
                before.split(beforeDelim).joinToString(separator = beforeDelim) { it.capitalize() }
            else
                before.capitalize()
```

```
        val newName = "$newBefore$delimiter$after" + if (file.extension.isEmpty()) "" else
".${file.extension}"
        val new = File("$dir$sep$newName")
        if (force)
            file.renameTo(new)
        return new
    }
```

Метод getReplacement – це метод, який з назви файлу повертає таблицю з шаблонами перейменування. Нижче представлений код функції.

```
fun getReplacements(): ArrayList<ArrayList<String>> {
    val out = ArrayList<ArrayList<String>>()
    translation.forEach { from, to -> out.add(arrayListOf(from, to, "translation")) }
    replacement.forEach { from, to -> out.add(arrayListOf(from, to, "replacement")) }
    replacementRegex.forEach { from, to -> out.add(arrayListOf(from, to, "regex")) }
    return out
}
```

У цьому класі також представлено багато споміжних методів, для коректної обробки даних. Методи clearReplacement, clearTranslation, clearRegex відповідають за видалення даних після обробки файлів.

Методи loadReplacement, loadTranslation, loadRegex – це інтерфейси для читання користувацьких шаблонів зміни з файлу.

Метод load – це власне реалізація трьох методів: loadReplacement, loadTranslation, loadRegex.

У цьому класі реалізований також метод rename. Цьому методу передаються файл, булева змінна та текст, який задає автор для подальшого перейменування файлу. В ньому міститься код, що відповідає за зміну назви. Також метод редагує українські або російські назви написані англійськими символами. До нього є два додаткові методи, які в подальшому використовує програма. Методи rename, translate викликають метод rename з потрібними параметрами, для досягнення потрібної цілі.

2.2.2. Опис класу FileChangerContainer

Цей клас представляє собою високорівневу обгортку над класом FileChanger.kt. Він відповідає за взаємодію частини інтерфейсу та обробку даних, робить обробку багато потоковою. Він містить такі основні методи: load, add, clear. Розглянемо кожен з методів детальніше.

Метод load – відповідає за виклик конкретної реалізації методу load з класу FileChanger. Цей метод читає конкретний список шаблонів замін з заданого файлу.

```
fun load(path: String, table: Type = Type.REPLACEMENT) {
    when (table) {
        Type.REPLACEMENT -> changer.loadReplacement(path)
        Type.TRANSLATION -> changer.loadTranslation(path)
        Type.REGEX -> changer.loadRegex(path)
    }
}
```

Метод add – відповідає за додавання нового шаблону у задану користувачем таблицю.

```
fun add(from: String, to: String, table: Type = Type.REPLACEMENT) {
    when (table) {
        Type.REPLACEMENT -> changer.addReplacement(from, to)
        Type.TRANSLATION -> changer.addTranslation(from, to)
        Type.REGEX -> changer.addRegex(from, to)
    }
}
```

Метод clear – відповідає за видалення елементів з таблиці. Цей метод також взаємодіє з класом FileChanger та відповідає за виклик конкретної реалізації методу clear з класу FileChanger.

```
fun clear(table: Type = Type.REPLACEMENT) {
    when (table) {
        Type.REPLACEMENT -> changer.clearReplacement()
        Type.TRANSLATION -> changer.clearTranslation()
        Type.REGEX -> changer.clearRegex()
    }
}
```

Також у цьому класі представлені інші методи з аналогічними властивостями. Усі вони спрямовані на взаємодію з класом FileChanger та на полегшення користування програми.

2.3. Модуль audiotag

Цей модуль складається з двох класів, які відповідають за обробку тегів аудіофайлів. Розглянемо кожен з класів, опишемо їх роботу, основні методи та взаємодію з іншими класами.

2.3.1. Опис класу TagCreator

Цей клас відповідає за генерацію метаданих у аудіофайлах на основі його імені в файловій системі та навпаки – імені файлу з його метаданих. Генерація відбувається за допомогою стандартного алгоритму або шаблону, який задає користувач. У ньому знаходяться такі основні методи: tagToName, nameToTag. Розглянемо кожен з перечисленних методів.

Метод tagToName – це реалізація генерації імені файлу з його метаданих. Генерація відбувається за стандартним алгоритмом або заданим користувачем шаблоном. Нижче представлений частковий код метода.

```
fun tagToName(file: File, pattern: String, force: Boolean,
              capitalize: List<Boolean>, ignore: Boolean,
              delimiter: String): File {
        …
    …
      val artist = tags.getOrDefault(tr.artist, "")
      val title = tags.getOrDefault(tr.title, "")
      val album = tags.getOrDefault(tr.album, "")
      val genre = tags.getOrDefault(tr.genre, "")
      val year = tags.getOrDefault(tr.year, "")
      var track = tags.getOrDefault(tr.track, "")
    …
    …
      }
      val new = pattern.replace("%a", artist)
            .replace("%y", year)
```

```
            .replace("%m", album)
            .replace("%g", genre)
            .replace("%n", title)
            .replace("%t", track)
        val path = file.absoluteFile.parentFile.absolutePath
        val extension = if (file.extension == "") "" else "." + file.extension
        val newFile = File(path + separator + new + extension)
    …
```

Метод nameToTag – відповідає за генерацію метаданих у аудіофайлі, на основі його імені в файловій системі. Генерація відбувається за стандартним алгоритмом або заданим користувачем шаблоном. Нижче представлений текст програми.

```
fun nameToTag(file: File, pattern: String, force: Boolean, capitalize: List<Boolean>, delimiter: String): Map<String, String> {
…
            if (pattern[i] == '%' && pattern[i + 1] in arrayOf('a', 'm', 'y', 'g', 'n', 't')) {
                i += 2
                if (i >= pattern.length)
                    value = name.substring(j)
                else
                    while (name[j] != pattern[i])
                        value += name[j++]
            }
            if (matcher.find())
                when (matcher.group()) {
                    "%a" -> artist = value
                    "%g" -> genre = value
                    "%m" -> album = value
                    "%n" -> title = value
                    "%t" -> track = value
                    else -> year = value
                }
            if (i >= pattern.length)
                break
        }
        if (capitalize[7])
            genre = genre.split(delimiter).joinToString(delimiter) { it.capitalize() }
        else if (capitalize[3])
            genre = genre.capitalize()
        if (capitalize[6])
            album = album.split(delimiter).joinToString(delimiter) { it.capitalize() }
        else if (capitalize[2])
```

```
        album = album.capitalize()
…
…
    if (artist.isNotEmpty()) tags.put(tr.artist, artist)
    if (album.isNotEmpty()) tags.put(tr.album, album)
    if (genre.isNotEmpty()) tags.put(tr.genre, genre)
    if (year.isNotEmpty()) tags.put(if (tags["version"] == "4") tr.date else tr.year, year)
 …
```

2.3.2. Опис класу TagReader

У цьому класі представлені методи для зчитування, зберігання та обробки тегів. Його основні методи: readTags, writeTags, FileInputStream.readFrame, getText. Розглянемо кожен з перечисленних методів.

Метод readTags – відповідає за зчитування з файлу метаданих для демонстрації на екран та подальшої обробробки.

Метод writeTags – після того як користувач вказує дані та зробить запит на зберігання потрібних даних, зберігає у заданий файл. У нього входять основні перевірки на існування заданого файлу, перевірка перед зберіганням на зміну метаданих та кількість зміни.

Метод FileInputStream.readFrame - відповідає за зчитування тексту з інтерфейсу користувача для подальшої обробки метаданих методами readTags, writeTags.

Метод getText – це функція, яка співставляє вказані символи користувачем із доступними для вводу. Цей метод є важливим тим, що для обробки метаданих використовують зарезервовані символи і їх не можна вказувати у полях тегів аудіофайлів.

2.4. Модуль setting

Цей модуль відповідає за налаштування програми а також включає налаштування, які задає користувач. Опишемо класи модуля та функції за які відповідає кожен клас.

2.4.1. Опис класу Language

Цей клас забезпечує підтримку багатомовного інтерфейсу користувача. У цьому класі такі основні методи: initLang, getLang, saveLang. Розглянемо кожен з перечисленних методів.

Метод initLang – відповідає за мову інтерфейсу при відкриванні програми. Він зчитує з інший файлів налаштувань мову інтерфейсу, яка вибиралась останній раз користувачем. Розглянемо кожен з перечисленних методів.

Метод getLang – використовується для інтерфейса користувача при зміні налаштувань мови.

Метод saveLang – використовується для зручності користувачу, він надає змогу йому змінити мову інтерфейса, з допомогою візуального списку користувач вибирає одну з доступних мов. Метод активує вибрану мову.

Основний код для зміни налаштувань мови інтерфейсу представлено нижче.

```
fun getLang(key: String): String = lang.getOrPut(key, {key})

fun saveLang() {
   val langName = Settings.getProperty("langActive")
   val filePath = languages[langName] ?: return
   Settings.saveFile(filePath, lang)
}

fun getLanguages(): Array<String> = languages.keys.toTypedArray()
fun getLangName(): String = Settings.getProperty("langName") ?: defaultLang
fun setLangName(value: String) {
   if (languages.containsKey(value))
      Settings.setProperty("langName", value)
}
```

2.4.2. Опис класу Force

У цьому класі зберігаються основні змінні з налаштуваннями для початкової роботи програми. Ці змінні може редагувати користувач при

роботі, програма автоматично їх завантажує при подальшій роботі. Основні методи, що реалізовані у класі: getMode, setMode. Розглянемо кожен з перечисленних методів.

Метод getMode – використовується при запуску програми. Він надає основні налаштування програмі такі як: мова інтерфейсу, розмір вікна та інше.

Метод setMode – викликається автоматично, при зміні основних налаштувань, зберігає дані для наступної роботи з програмою.

2.4.3. Опис класу SaveLoad

Клас містить два методи: getSaveLoad, setSaveLoad. Він відповідає за зберігання даних пов’язаних з зберіганням, зміною, обробкою, переміщенням, видаляннням, створеннням файлів під час роботи з програмою. Розглянемо кожен з перечисленних методів.

Метод getSaveLoad – завантажує усі необхідні дані з файлової системи користувача.

Метод setSaveLoad – зберігає усі зміни у файлову систему користувача.

2.4.4. Опис класу SizeResolver

Клас що відповідає за розмір вікна програми. Він містить три метода: getSize, setSize, checkKey. Розглянемо кожен з перечисленних методів.

Метод getSize – відповідає за вибір та встановлення розміру вікна робочої програми, за умови що це не початкове завантаження програми. Нижче представлений код метода.

```
fun getSize(key: String): Int? {
    if (checkKey(key))
        return Settings.getProperty(key)?.toIntOrNull()
    return null
}
```

Метод setSize – це метод, що відповідає за встановлення вікна робочої програми заданих розмірів. Нижче представлений код метода.

```
fun setSize(key: String, value: Int) {
    if (checkKey(key))
        Settings.setProperty(key, value.toString())
}
```

Метод checkKey – відповідає за початкові налаштування вікна робочої програми, використовує дані з інших класів налаштувань.

2.4.5. Опис класу Settings

Основний клас налаштувань, саме він підтягує дані з інших класі налаштувань та надає доступ до них. Основні методик класу: loadFile, saveFale, getProperty, setProperty, getDirectory. Розглянемо кожен з перечисленних методів.

Метод loadFile – відповідає за завантаження файлів в поточній папці у спеціальну частину інтерфейсу, яка у вигляді таблиці. Це полегшує подальшу роботи з файлами.

Метод saveFale – основний метод, шо відповідає за зберігання файлів після обробки метаданих файлу.

```
fun saveFile(filePath: String, table: HashMap<String, String>) {
    val path =
        if (filePath.matches("^(/|[A-H]:\\\\).*".toRegex()))
            filePath
        else
            directory + filePath
    OutputStreamWriter(FileOutputStream(path), lang.defaultCharset).use {
        table.forEach { key, value -> it.appendln("$key$csv$value") }
    }
}
```

Метод getProperty – відповідає на запит про зміну налаштувань та надає усі загальні налаштування на даний момент, для подальшого редагування їх.

Метод setProperty – зберігає усі загальні налаштування, які були змінені.

Метод getDirectory – метод, який відповідає на запит та повертає шлях до поточної папки.

2.5. Модуль gui

Цей модуль відповідає з графічну частину програму. Він містить в собі додатковий модуль та класи. Розглянемо їх роботу детальніше.

2.5.1. Опис класу TableView

Цей клас містить один метод init, у ньому задається мінімальний розмір таблиці. Ця таблиця містить у собі всі файли поточної папки та метадані. Також у ньому міститься реалізація повзунка, для полегшення роботи користувачу.

2.5.2. Опис класу TableColumnManager

Клас для зручного користування таблиці, він використовує бібліотеку Java для навігації таблиці. Він надає можливість користувачу задавати за допомогою мишки потрібної ширини колонок, рядків. Автоматично генерує потрібний розмір рядків, стовбців. Також переналаштовує параметри в залежності від розміру таблиці. Дозволяє записувати нові дані для подальшої обробки чи зберігання метаданих. Реалізує панель додаткову при кліканні правою клавішею миші та інші налаштування для зручної роботи користувача з таблицею.

2.5.3. Опис класу MainView

Клас реалізує візуалізації головного вікна інтерфейсної частини. Його основними методами являються: close, load, quit, show, createFolder, deleteFolder. Розглянемо кожен з перечисленних методів.

Метод close – відповідає за закриття основної сторінки та за зберігання даних після роботи на цій сторінці. Для зберігання інформації він використовує метод quit, який буде описаний нижче. Код метода:

```
fun close() {
    if (Settings.getForce("forceQuit")) {
        quit()
        return
    }
    val result = JOptionPane.showConfirmDialog(this, Settings.getLang("Quit?"), "",
JOptionPane.YES_NO_OPTION)
    if (result == JOptionPane.YES_OPTION) {
        quit()
    }
}
```

Метод load – будує модель сторінки та запитує про налаштування в інших методів, для подальшої візуалізації сторінки. Нижче представлений код метода.

```
private fun load() {
    if (Settings.getSaveLoad()) {
        model.loadFiles(Settings.getSaveLoadPath())
    }
    table.load()
}
```

Метод quit – викликає методи, які відповідають за зберігання даних. Викликає методи для зберігання даних в таблиці, для зберігання шляхів папок та файлів та інших. Нижче представлений код метода.

```
private fun quit() {
    if (Settings.getSaveLoad())
        model.saveFiles(Settings.getSaveLoadPath())
    table.save()
    Settings.saveLang()
```

```
    dispose()
  }
```

Метод show – реалізує візуалізацію за моделлю, яка створюється у методі load.

```
fun show(dir: File) = model.showDir(dir)
```

Метод createFolder – створює папку у заданій директорії та перемальовує таблицю з урахуванням цієї папки.

Метод deleteFolder – видаляє задану папку та перемальовує таблицю без цієї папки, можливе видаляння одночасно декількох папок.

2.5.4. Опис класу FileTree

У цьому класі реалізовані основні методи для представлення файлової структури користувача. У ньому містяться такі основні функціїї: getSelected, setRoot, update, class FileTreeModel. Розглянемо кожен з перечисленних методів.

Метод getSelected – реалізує вибір певної папки та відображення її у таблиці. Нижче представлений код метода.

```
  fun getSelected(): List<File> {
    return tree.selectionPaths?.map { (it.lastPathComponent as
DefaultMutableTreeNode).userObject as File } ?: ArrayList()
  }
```

Метод setRoot – реалізує частину інтерфейсу, де візуалізовується файлова система у вигляді дерева. Нижче представлений код метода.

```
  fun setRoot(file: File) {
    tree.model = FileTreeModel(file)
  }
```

Метод update – оновлює візуалізацію файлової системи, викликається метод після змін у ній. Нижче представлений код метода.

```
fun update() = (tree.model as FileTreeModel).reload()
```

Внутрішній клас FileTreeModel – відповідає за повну реалізацію файлової системи у вигляді дерева, це зроблено для подальшої зручної візуалізації файлової системи користувача.

2.5.5. Опис класу FileTable.kt

Цей клас зображує таблицю. У ньому є метод що відповідає за виділенням вибраних файлів в поточній папці, для подальшої обробки. До нього входять такі основні методи: save, load. Розглянемо кожен з перечисленних методів.

Метод save – відповідає за зберігання даних, після обробки у поточній директорії. Код метода:

```
fun save() {
    FileWriter(path).use { writer ->
        columnModel.columns.toList().forEach {
            val index = it.modelIndex
            writer.appendln("$index$csv${it.preferredWidth}$csv${tcm.isHidden(index)}")
        }
    }
}
```

Метод load – відповідає за відображення таблиці з файлами вибраної директорії.

2.5.6. Опис класу FileModel.kt

Цей клас реалізує модель поведінки файлової системи, для представлення структури даних у виді дерева. Його основні методи: addFile, saveFiles, loadFiles, getTags, select, selectName, selectExt, save. Розглянемо детальніше кожен з перечислених методів.

Метод addFile – відповідає створює новий файл в файловій системі користувача та записує його до таблиці.

Метод saveFiles – відповідає за зберігання файлу у файловій системі користувача. Нижче наведений код методу.

```
fun saveFiles(path: String) {
        val file = File(path)
        val dir = file.absoluteFile.parentFile
        if (!dir.exists())
            dir.mkdirs()
        FileWriter(file).use { writer -> data.forEach { writer.appendln((it.last() as
File).absolutePath) } }
    }
```

Метод loadFiles – відповідає за відкриття файлув таблиці з демонстрацією тегів метаданих.

```
    fun loadFiles(path: String) {
        val file = File(path)
        if (file.exists()) {
            FileReader(file).use {
                it.forEachLine {
                    val new = File(it)
                    if (new.exists())
                        addFile(new)
                }
            }
        }
        fireTableDataChanged()
    }
```

Метод getTags – відкриває і демонструє теги метаданих для подальшого редагування аудіофайлів.

Метод select – викликається при «подвійному кліці» мишкою на потрібний рядок таблиці з файлом. Тоді цей файл вибирається для подальшої обробки.

```
    fun select(filter: List<String>, ext: Boolean) {
        val regex = filter.map { Regex(it) }
        if (ext)
            selectExt(regex)
        else
            selectName(regex)
        fireTableDataChanged()
    }
```

Метод selectName – при «подвійному кліці» мишкою на назву файлу викликається ця функція. Після цього програма очікує зміни назви файлу.

Метод save – відповідає за зберігання аудіофайлів після обробки полів(тегів) метаданих.

2.5.7. Підмодуль toolbar

Клас ConvertTab – цей клас відповідає за графічну реалізацію генерацію метаданих у аудіофайлах, на основі вибраного імені в файловій системі та навпаки – імені файлу з його метаданих. У ньому знаходяться два методи. Розглянимо основний його метод детальніше.

У методі init реалізовані кнопки, випадаючі списки з повзунками, радіокнопка для встановлення флага спеціальні поля для вводу та редагування аудіофайлів. Кожна з цих частин є частинкою великого механізму для обробки тегів метаданих.

Клас FilesTab - – це клас, що являється компонентом панелі інструментів. Він відповідає за зображення роботи файлів в файловій системі такої як: навігація, створення, видалення папок та вибір одного чи декількох файлів для наступної обробки. Він має в реалізації тільки один метод.

Цей метод створений для управління візуальною системою, він розміщує в інтерфейсі іконки на кнопки для реалізації інтуїтивно зрозумілого інтерфейсу та задає розмір іконок і кнопок. А також при натисканні на кнопку викликає потрібний метод. Нижче продемонстровано код однієї з багатьох дій цього методу.

```
val create = LIcon("createFolder.png")
    create.addActionListener { MainView.createFolder() }
```

Клас FilterTab – це клас, що відповідає за візуалізацію пошуку по усім наявним файлам за заданими шаблонами або фільтрами. Основними елементами цього класу являються: run, class TableModel, add, delete, get. Детальніше розглянемо кожен елемент.

Клас TableModel – додатковий внутрішній клас, завдяки якому по спеціальній модель під час пошуку проходить програма, беручи до уваги задані фільтри.

Метод add – додає новий файл під час пошуку користувачем, всі зміни зберігаються після натискання кнопки, що реалізує збереження змін.

Метод delete – відповідає за видалення списку елементів вибраних при пошуку за заданим фільтром. Нижче представлений код метода:

```
fun delete(rows: List<Int>) {
    rows.sortedDescending().forEach { data.removeAt(it) }
    fireTableDataChanged()
    if (save) save()
}
```

Метод get – відповідає за пошук елемента по шаблону, подіє настає коли користувач натисне відповідну кнопку.

```
fun get(): List<String> = data.filter { it[0] == true }.map { it[1] as String }
```

Клас ReplaceTab – це клас, що реалізує графічну частину, що прямо пропорційно взаємодіє з процесом зв'язаним з перейменуванням, видаленням та перезавантаженням файлів користувача.

У класі прописаний вигляд таблиці для редагування файлів, кнопок, які відповідають за дії над файлами, стилістика таблиці, панелі, кнопок та реалізація іконок на кнопках.

Клас SettingsTab – це клас, що відповідає за зображення вікна налаштувань проекту. У класі реалізовані налаштування розміру вікна, мова інтерфейсу, розміра та стилістика блоків, що знаходяться у вікні. Також у ньому описується дії при закритті проекту, зберігання налаштувань які не повинні змінитись при наступному запуску, ще одне зберігання усіх файлів та запити про закриття програми та зберігання даних.

Інтерфейс TagsTab – являється інтерфейсом для подальшого наслідування класами. Він надає доступ до обробки тегів метаданих аудіофайлів. У ньому прописані змінні полів тегів, які в подальшому

обробляються та утворення найменших блоків вікна інтерфейсу. Нижче приведено частина коду з нього.

```
    private val title = JTextField()
    private val artist = JTextField()
    private val album = JTextField()
    private val year = JTextField()
    private val track = JTextField()
…
…
    panel.add(LLabel("Album"))
    panel.add(album)
```

Клас ToolBar.kt – це клас, що реалізує зображення вікна зі списком поточних файлів та панелі налаштувань. Він відповідає за основні блоки в яких находяться таблиці кнопки та інше, задає мінімальний їх розмір. Реалізує візуалізацію кожної з панелей та надає назви кожній з них. Нижче приведено частина коду з нього.

```
        init {
            layout = BoxLayout(this, BoxLayout.X_AXIS)
            minimumSize = Dimension(600, 140)

            add(FilesTab())
            add(tabs)

            addTab("Filter", FilterTab())
            addTab("Replace", ReplaceTab(false))
            addTab("Translate", ReplaceTab(true))
            addTab("Tags", tagsTab)
            addTab("Convert", ConvertTab())
            addTab("Settings", SettingsTab())
        }
```

3. ТЕСТУВАННЯ КОМПОНЕНТІВ СИСТЕМИ

3.1 Підхід до тестування системи

Для тестування роботи різних компонентів існує багато підходів. Тестування роботи даної програми буде виконуватись за допомогою методу модульного тестування. Модульне (unit) тестування – це метод тестування програмного забезпечення, який полягає в окремому тестуванні кожного модуля коду програми. Модулями для тестування вважають мінімальну частину програми, яку можна протестувати.

Модульне тестування в мові Java виконується за допомогою бібліотеки JUnit. Дана бібліотека надає наступні можливості:

- можливість попередньої ініціалізації усіх змінних спільних для усіх тестових ситуацій;
- можливість виводу інформації про хід тестування в текстовому чи графічному вигляді;
- можливість декомпозиції структури тестового покриття;
- можливість запуску декількох тестів паралельно;
- можливість
- та ін.

Також важливим аспектом тестування є інтерфейсна частина. Так як за допомогою бібліотеки Junit не можливо провести тестування, воно буде проводитись вручну. У тестування інтерфейсу входять такі аспекти:

- тестування усіх елементів та блоків інтерфейсу
- тестування візуального розташування компонентів
- тестування зручності використання

3.2 Тестування інтерфейсу програми

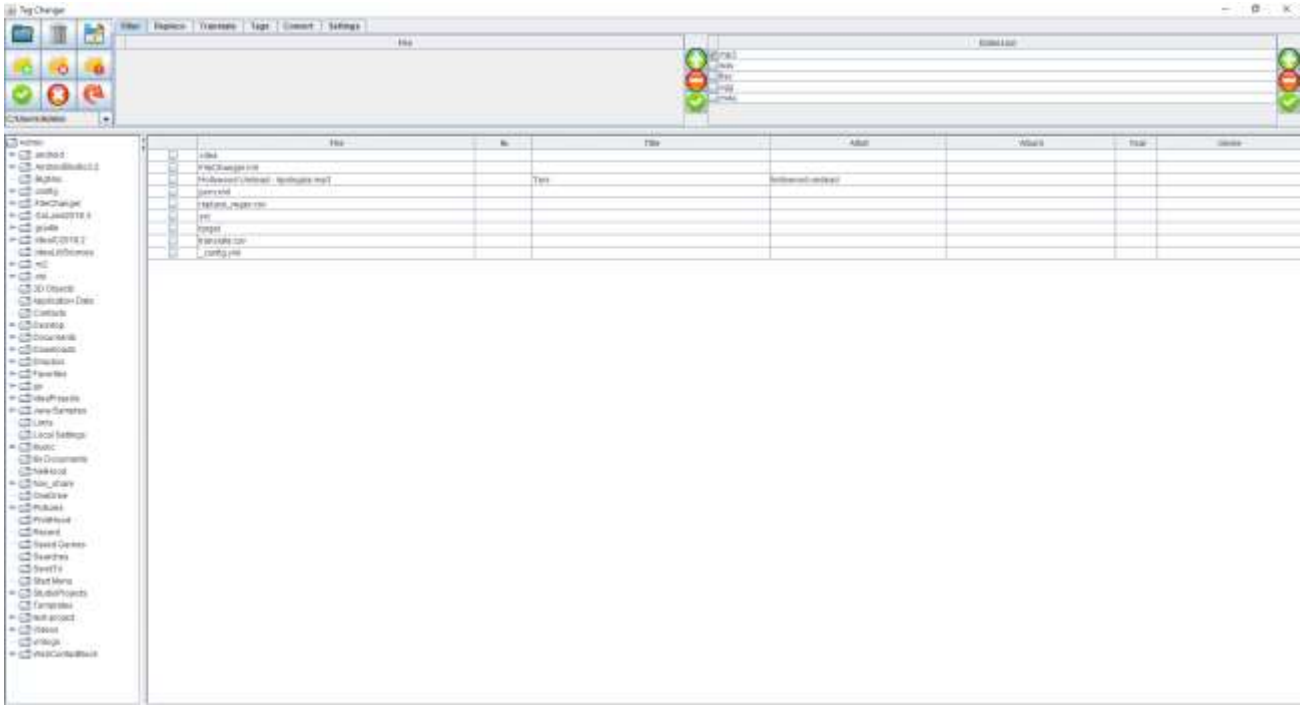

Рис. 9. Інтерфейс програми

Інтерфейс програми, зображений на рисунку 9, виконаний у мінімалістичному стилі. Для зображення всіх елементів та не нагромадження інтерфейсу було вибрано панель зі зміними вкладками. Ця панель розташовується зверху та є інтуїтивно зрозумілою. В інтерфейсі реалізовані кнопки з іконками, які полегшують навігацію по програмі. Розглянемо та протестуємо кожну з частин інтерфейсу.

Для роботи з файловою системою користувача було виділено окремий блок з правої сторони. Коли користувач вибрав потрібну папку для редагування файлів, потрібна директорія зображується в окремому блоці. Цей блок знаходиться по центі. Під нього виділяється найбільшу частину інтерфейсу програми, бо саме з цим блоком в подальшому користувач буде взаємодіяти для редагування файлів.

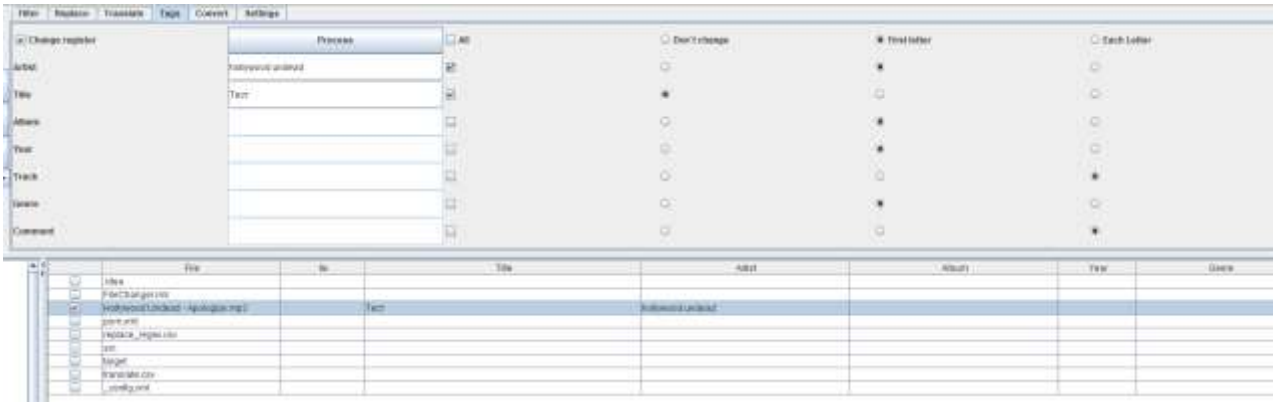

Рис. 10. Інтерфейс частини для редагування метаданих

На рисунку 10 зображена частина інтерфейсу, яка призначення для редагування тегів аудіофайлів. У цій вкладці представлені основні теги для редагування аудіофайлів. Вибравши потрібні файли для редагування з таблиці, користувач задає потрібні зміни радіокнопками та кнопками-перемикачами. Також з рисунку видно, що є спеціальні рядки, в яких користувач вводить потрібні дані. Після всіх потрібних виконаних дій потрібно натиснути кнопку «Process», тоді всі зміни записуються в файл і їх зразу можна побачити в таблиці з файлами, оскільки там є колонки, які відповідають за метадані.

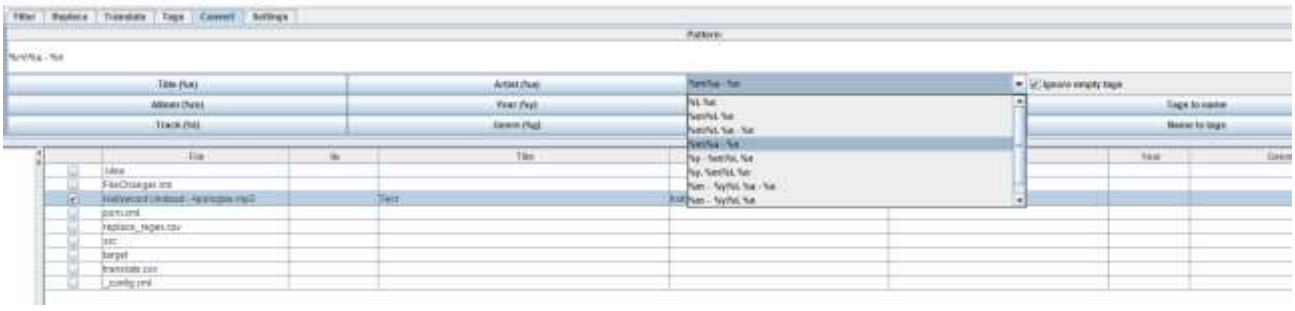

Рис. 11. Інтерфейс для шаблонів аудіофайлів

У інтерфейсі реалізовано два варіанта генерації шаблонів для аудіофайлів, вони представлені на рисунку 11. Перший полягає у тому, щоб вибрати потрібний шаблон з випадаючого списку. Іншим методом є задавання користувачем шаблону. Це можна зробити за допомогою кнопок. На кожній кнопці є надпис з тегом. Після вибору шаблону користувач

повинен нажати кнопку «Name to tags». Також у цьому вікні інтерфейса можна записувати інформацію в аудіофайли за шаблоном, який був раніше вибраний. За це відповідає кнопка «Tags to name».

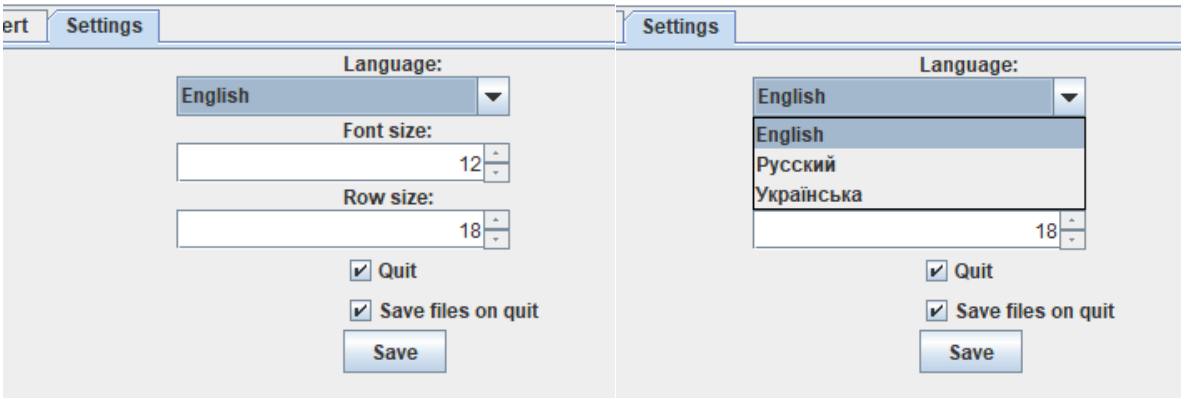

Рис. 12. Інтерфейс налаштувань

На рисунку 12 зображений інтерфейс з вкладкою налаштувань. Користувач обирає мову з заданого випадаючого списку. Також можна задати розмір шрифта в таблиці та розмір рядків таблиці за допомогою маленьких кнопок або вручну ввести потрібну цифру. Для зручності багаторазового користування є дві радіокнопки. Кнопка «Quit» - відповідає за вікно з запитом про вихід, бо користувач може ненароком натиснути. Кнопка «Save files on quit» - відповідає за збереження файлів та змін при виході з програми.

Рис. 13. Інтерфейс редагування файлів

На рисунку 13 зображений інтерфейс для перейменування файлів. З права на рисунку є кнопки для роботи з файлами, для зміни файлів треба

натиснути кнопку «+». У таблицю добавляється новий рядок для зміни ще одного файлу, вводимо назву файлу або шлях до нього у поле в колонці «from» та бажану назву до колонки «to». Після цього натискаємо зберегти для зберігання фалу і продовження аналогічних дій або натискаємо галочку для закінчення роботи з перейменуванням файлів.

3.3 Тестування модулів програми

Тестування програми було одразу розбите на 3 основних модуля:

- FileChangerContainerTest
- FileChangerTest
- TagReaderTest

Кожен клас для тестування програми містить методи для перевірки коректності роботи основних методів відповідного класу у програмі. Результатом виконаної роботи кожного метода є слова passed, warning, error. У кожному методі задаються початкові параметри і викликається відповідна програма, а потім порівнюємо з правильною відповіддю. Нижче представлений уривок коду для представлення вигляду тестування.

```
@Test
fun translateNoExtensionTest() {
    val file = File("testoVa prograMa")
    file.createNewFile()
    val changer = FileChanger()
    changer.loadTranslation()

    val new = changer.translate(file)

    assertTrue(new.exists())
    assertEquals("тестоВа програМа", new.name)
    assertFalse(file.exists())

    new.delete()
}
```

Клас FileChangerTest провіряє основні зміни зв’язані з об’єктом, такі як: перейменування, переміщення файлів та папок, перевірка правильності перейменування за шаблоном при трансліті (приклад: «Imya» - «Ім’я»), перевірка перейменування з урахування регустру (приклад: «nAmE» - «Name»). Він містить такі тести: translateNoExtensionTest, translateExtensionTest, renameTest, moveTest, moveAbsolutePathTest, moveByNameToTest, moveByNameFromToTest, moveByNameTest, renameRegexTest, directoryRenameTest.

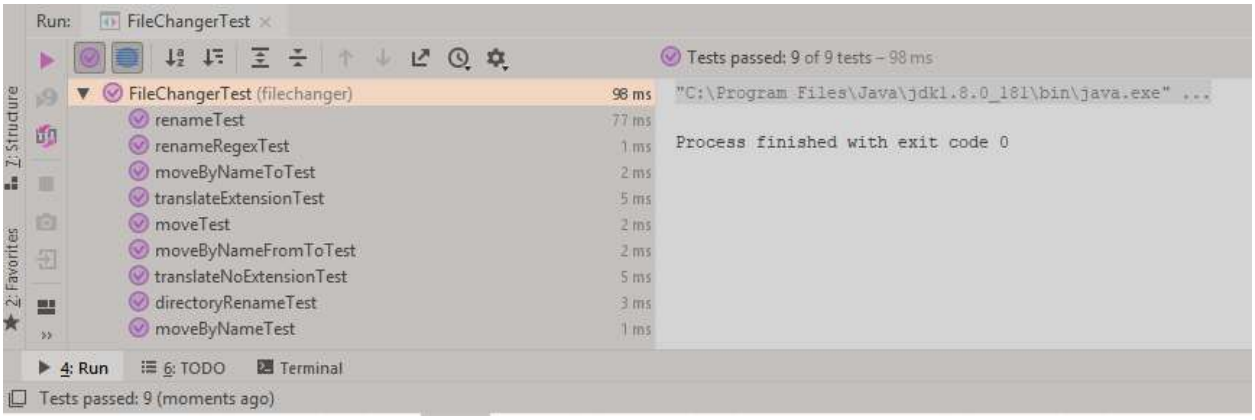

Рис. 14. Робота FileChangerTest

На рисунку 14 продемонстровано виконану роботу, кожен з методів класу FileChanger пройшов перевірку успішно.

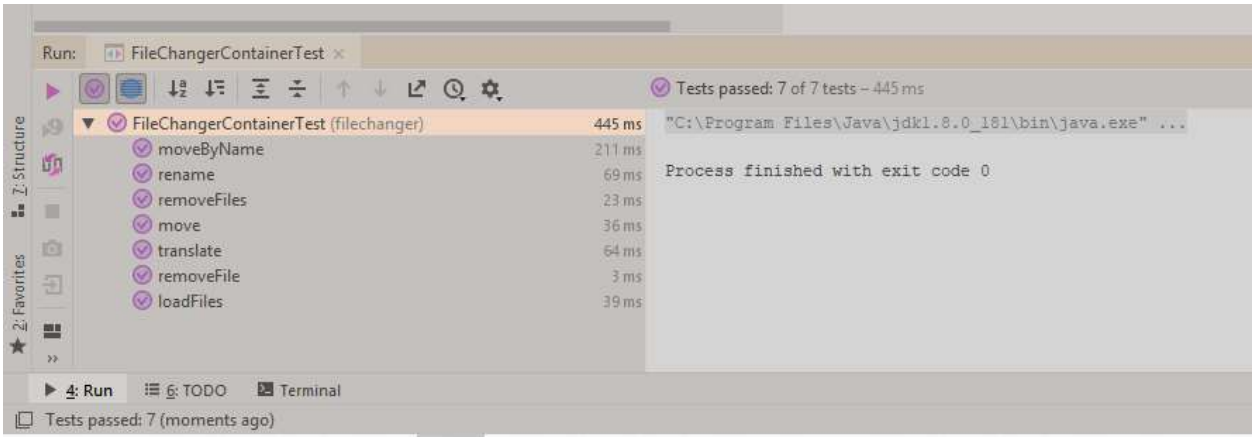

Рис.15. Робота FileChangerContainerTest

Клас FileChangerContainerTest провіряє багатопотокову обробку та вибір потрібного метода в залежності від введених параметрів. Він містить такі тести: rename, translate, move, moveByName, loadFiles, removeFile, removeFiles. Кожен з класів пройшов перевірку правильно, це продемонстровано на рисунку 15.

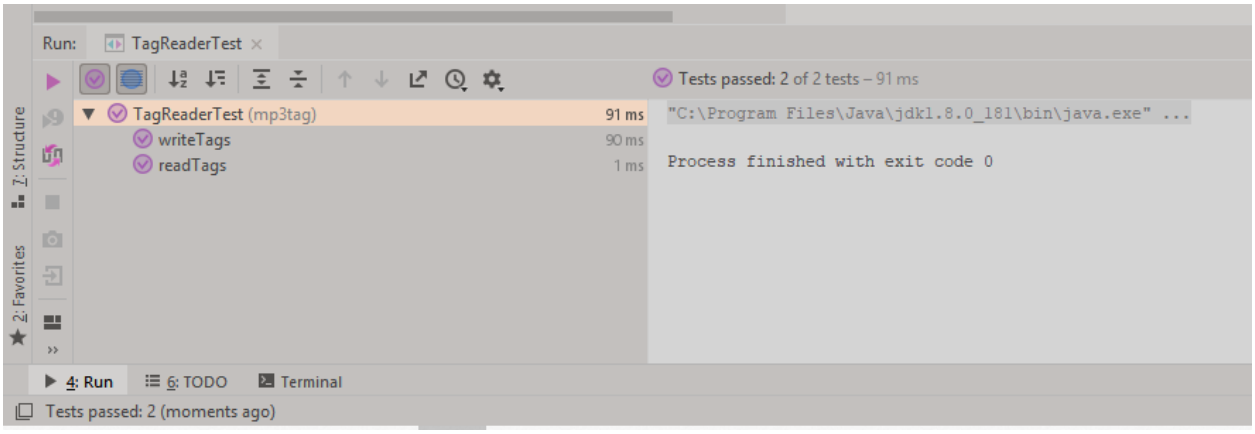

Рис.15. Робота TagReaderTest

Клас TagReaderTest провіряє коректність витягнення та запис тегів з аудіофайлів. Містить такі тести: readTags, writeTags. Клас та його методи пройшли перевірку успушно.

4. АНАЛІЗ, ПОРІВНЯННЯ ПРОГРАМ

4.1 Порівняння програм

Програми для порівняння вибиралась тільки ті, що мають інтерфейс для взаємодії з користувачем. Проаналізуємо представлені на ринку програми для обробки аудіофайлів, розберемось у їх перевагах та недоліках при роботі, цілях та доречності використання кожної з програм, а також порівняємо з власною розробленою програмою. Аналіз програм базується на власному досвіді використання цих програм та зборі інформації від користувачів даних програм та статей про них.

Для наглядності характеристики розподілені на 3 групи та представлені у порівняльних таблицях.

Назва програми	Зручність використання	Спрямування програми	Тип ОС	Ліцензія	Залежність від додаткових бібліотек
Metatogger	-	Для професійного використання	Windows, Linux, OS	Платна	Залежна від зовнішніх бібліотек
TigoTago	+	Для персонального використання	Windows	Безкоштовна	Незалежна
Music Brainz Picard	-	Для персонального використання	Windows, Linux, OS	Безкоштовна	Незалежна
Tag Scanner	+	Для професійного використання	Windows	платна	Незалежна
MP3 tag	+	Для персонального використання	Windows, Linux, OS	платна	Незалежна

Розроблена програма	+	Для персонального використання	Windows, Linux, OS	Безкоштовна	Незалежна

Табл. 4. Порівняння програм за використанням.

Для початку проаналізуємо усі представлені програми за спрямуванням. Для професійного використання призначені дві програми, інші – для персонального використання. Більшість програм підтримують усі ОС. TigoTago і Tag Scanner працює тільки на ОС Windows, але на противагу іншим вони безкоштовні. Усі інші програми являються плантними. Також вагомим недоліком програми Metatogger є те, що вона не працює без додаткових фреймворків. Усю цю інформацію детальніше можна побачити на таблиці 4.

Назва програми	Інтерфейс	Багатомовність	Змінні налаштування інтерфейсу
Metatogger	Не зручний	-	+
TigoTago	Зручний	-	+
Music Brainz Picard	Не зручний	+ (англ., франц.)	+
Tag Scanner	Не зручний	-	+
MP3 tag	Зручний	-	-
Розроблена програма	Зручний	+ (англ., укр., рос.)	+

Табл. 5. Порівняння програм за інтерфейсом.

Інтерфейси програм Metatogger, Music Brainz Picard, Tag Scanner є досить не зручні для використання. Metatogger і Tag Scanner спрямовані на професійне використання, тому складність інтерфейсу пояснюється великим набором інструментів. Багатомовна підтримка є тільки у власній програмі та Music Brainz Picard. Налаштування для зміни шрифту тексту, зміни розмірів таблиць та інше є у всіх крім MP3 tag. Усю цю інформацію детальніше можна побачити на таблиці 5.

Назва програми	Справність роботи	Робота з файловою системою	Підтримка форматів	Робота з файлами
Metatogger	Задовільна	+	Більшість форматів	Поодинока
TigoTago	Задовільна	-	Аудіо та відео форматів	Групова, поодинока
Music Brainz Picard	Задовільна	-	Основні формати	Групова, поодинока
Tag Scanner	Задовільна	+	Більшість форматів	Групова, поодинока
MP3 tag	Незадовільна	-	Основні формати	Поодиноке
Розроблена програма	Задовільна	+	Основні формати	Групова, поодинока

Табл. 6. Порівняння роботи програм

У програмі MP3 tag є декілька функцій, які не працюють і це є великим недоліком. Усі інші програми добре виконують зазначені функції. Роботу з файловою системою користувача мають тільки 2 програми. Цей фактор є

досить важливим при груповій обробці, бо деколи потрібно створити зразу папку чи перенести в іншу. Більшість людей користуються основними форматами, тому для особистого використання підходять усі програми. Деякі програми підтримують велику кількість форматів і TigoTago навіть відео формати, але це потрібно тільки для професійної роботи. Також значною перевагою є групова обробка аудіофайлів, це значна перевага для пришвидшення роботи. Наприклад, коли потрібно замінити автора для всіх його композицій, а їх може бути пару десятків в колекції. Детальніша інформація представлена в таблиці 6.

4.2 Аналіз результатів

Проаналізувавши програми, очевидно, що кожна з них має певні переваги та недоліки. Це переважно зумовлено спеціалізацією програм на кожних форматах, масштабах обробки даних, ретельності роботи з ними.

Деякі програми роблять акцент на професійному використанні, як наслідок, вони мають складний інтерфейс і багато інструментів для обробки аудіофайлів. Ці програми є потребують використання багатьох ресурсів і потрібними у деяких галузях роботи з аудіоконтентом. Для простого користувача вони занадто складні у використанні і більшість інструментів не потрібні.

Інші програми мають з простий інтерфейс та мінімально необхідний набір інструментів для роботи з аудіофайлами. Їх недоліками зазвичай стають неможливість редагування одночасно декількох аудіофайлів або не можливість взаємодії аудіофайлів з файловою системою комп’ютера, на якому запущена дана програма. Також проблемою деяких з програм є те, що вони реалізовані під якусь одну операційну систему.

Кожна програма розроблена відповідно до певних потреб користувачів та які і є її цільовою аудиторією.

Серед можливостей розробленої програми є не тільки обробка метаданих аудіофайлів, але й легке управління колекціями аудіофайлів. Вона надає можливість кожному користувачу переносити, видаляти та добавляти файли різних типів та в будь-якій кількості. Недоліками реалізованої програми є підтримка тільки найпопулярніших форматів. Причиною цього недоліку є мінімізація використання ресурсів програми, підтримка лише найпопулярніших форматів є доцільною, так як всі інші формати, я к правило, є високоспеціалізованими і використовуються лише на професійному рівні роботи з аудіоконтентом. Іншим недоліком є неможливість пошуку метаданих аудіофайлів у Інтернеті. Розроблена програма має набір необхідних інструментів для редагування метаданих колекції аудіофайлів користувача.

ВИСНОВКИ

Розроблена програма дозволяє користувачам легко управляти файловою системою на комп’ютері. Основною частиною програми є саме спрощення роботи з колекціями аудіофайлів. Програма дозволяє людині, яка не розбирається в структурі аудіофайлів швидко змінити теги метаданих. Також для зручного користування важливим аспектом став доступ до обробки в файловій системі. Користувач може легко вибирати потрібні аудіофайли та швидко перекинути їх у нову папку. Програма була розроблена з врахуванням аналогів які представлені на ринку та спрямована на конкретних користувачів. Програма містить мінімальний набір інструментів для елементарної обробки файлів, для більш складної та професійної діяльності є відповідні програми з великим набором користувацьких інструментів.

Програма спрямована на категорію людей, які хочуть обробляти, впорядковувати власну колекцію аудіофайлів та використовувати як базову програму для обробки метаданих.

Переваги: можливість застосовувати на різних операційних системах; простота у використанні та полегшена робота з впорядкуванням колекцій аудіофайлів; можливість роботи з файловою системою; наочність результатів обробки.

Недоліки: підтримка тільки найпопулярніших форматів аудіофайлів.

Можливі модифікації:

- додавання нових інструментів для управління аудіофайлами;
- додавання більш красивого дизайну інтерфейсу;
- додавання інших форматів аудіофайлів.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Симонович С. В. Информатика. Базовый курс. Учебник для вузов / Симонович С. В. и др. — СПб.: Издательство Питер, 1999. — 640 с.
2. Копанєва В. О. Формати опису мережевих інформаційних ресурсів / В. О. Копанєва // Документознавство. Бібліотекознавство. Інформаційна діяльність: Проблеми науки, освіти, практики: Зб. матеріалів VIII Міжнар. наук.-практ. конф., Київ, 17-19 травня 2011 р. — К., 2011. — С. 187–189.
3. Електронний ресурс: https://web.archive.org/web/20120620142716/http://www.unixgods.org/~tilo/ID3/docs/ID3_comparison2.html
4. Прокимнов Н. «Инфраструктура информационной системы мониторинга экономических процессов» 2011. — С. 67–83.
5. Nikil J., Johnston J., Robert Safranek. (October 1992). «Signal Compression Based on Models of Human Perception». Proceedings of the IEEE. — С. 1385—1422.
6. Finlayson R. A More Loss-Tolerant RTP Payload Format for MP3 Audio — Internet Engineering Task Force, 2008. — 22 p.
7. Casner S., Hoschka P. «Type Registration of RTP Payload Formats» — Internet Engineering Task Force, 2003. — 45 p.
8. Nilsson M. The audio metadata Media Type — Internet Engineering Task Force, 2000. — 5 p.
9. Peek T., Hans B., The emergence of the compact disc, IEEE Communications Magazine, Jan. 2010, pp. 10–17.
10. Ecma International. Standard ECMA-130: Data Interchange on Read-only 120 mm Optical Data Disks (CD-ROM), 2nd edition (June 1996).